

Advantage™ VISION:Builder®

Advantage™ VISION:Two™ for

OS/390®

Getting Started Guide

14.0



Computer Associates™

BUGTO140.PDF/D92-009-014

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, the user may print a reasonable number of copies of this documentation for its own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software of the user will have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2002 Computer Associates International, Inc. (CA).

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: Introduction

Features.....	1-2
Efficient Database Access.....	1-2
Wide Variety of Report Output Formats.....	1-2
Migrating Data.....	1-2
Meta Data Repository System	1-2
Verifying the Data	1-3
Handling Complex Applications Easily	1-3
A Computer Associates Solution	1-3
Benefits.....	1-3
VISION:Builder Requirements.....	1-4
Environment	1-4
IBM Operating Systems Supported.....	1-4
Available Interfaces.....	1-4
SMP/E Installation Specifications	1-4
CA LMP Licensing Specifications.....	1-5
Documentation	1-6
Installing Online Documentation and the Acrobat Reader	1-11
Viewing Online Documentation	1-11
Educational and Professional Services.....	1-11
Contacting Total License Care (TLC).....	1-12
Contacting Computer Associates.....	1-13

Chapter 2: Enhancements and Modifications

General Enhancements	2-3
ASL Run Control Statements	2-3
ASL Report Statements	2-8
ASL EXTRACT Statement	2-11
Additional ASL Procedure Statements	2-12
Long Field Names	2-13
Remove Limit of 10 Extracted Data Files (Subfiles)	2-13
Extracted Data Files Following the Sort	2-14
Customer-Requested Enhancements	2-15
PL/I-Like Varchar Output	2-15
Delimited Data Output Enhancements	2-16
HTML Primary Document Name Change	2-18
HFS Output for HTML Report	2-18

Chapter 3: Installing VISION:Builder

Installation Tasks	3-1
--------------------------	-----

Chapter 4: VISION:Builder Quick Reference

Specialized Report Formats	4-1
Reporting from Special Data Files	4-2
Data Field and Record Processing	4-3
File Manipulation	4-4
Definition Processing	4-6
COBOL and VISION:Builder	4-7
Table Processing	4-7
Performance Tuning	4-7

Chapter 5: COBOL Quick Start

Flow Diagram	5-2
Utility Execution	5-3
Using CA-Panvalet and CA-Librarian COBOL Copybooks	5-4
CA-Panvalet Interface	5-4
CA-Librarian Interface	5-5
Control Statements	5-7
Coding Rules	5-7
FILEGEN Control Statement	5-7
SEGMENT Control Statement	5-9
\$COBOL and \$ECOBOL Control Statements	5-11

Conversion Rules	5-11
Generated COMLIB File Definition.....	5-11
COMLIB Field Name Generation	5-12
Unsupported COBOL Specifications.....	5-13

Chapter 6: DB2 Quick Start

Flow Diagram	6-2
Utility Execution.....	6-2
Control Statements.....	6-4
Coding Rules.....	6-5
DB2CNTL Control Statement.....	6-5
FILEGEN Control Statement	6-6
SEGMENT Control Statement.....	6-8
NEWPAGE Control Statement	6-9
Conversion Rules	6-9
Generated COMLIB File Definition.....	6-9
COMLIB Field Name Generation	6-10
Generating COMLIB Field Information.....	6-10

Chapter 7: VISION:Results Quick Start

Flow Diagram	7-1
Utility Execution.....	7-3
DD Statement Overrides	7-5
Operational Characteristics	7-5
Supported Statement Types	7-5
Converted File Definition	7-6
Member Naming Conventions.....	7-7
SYSPRINT Listing	7-7
COPYP and COPYL Support.....	7-7
Installing the VISION:Results Quick Start Routines (Optional)	7-8
Link Edit CA-Librarian Support	7-8
Link Edit CA-Panvalet Support	7-9
Messages.....	7-9
Return Codes	7-11

Chapter 8: VISION:Inquiry Quick Start

Flow Diagram	8-2
Utility Execution.....	8-3
FILEGEN Control Statement	8-4
Coding Rules.....	8-5
Syntax.....	8-5
Conversion Rules	8-6
Generated File Definition.....	8-6
Field Name Generation	8-7
Messages.....	8-7
Return Codes	8-9

Appendix A: ASL Examples

Code	A-1
Reports	A-2
Listing	A-3

Index

Note: Unless otherwise indicated, this book describes both VISION:Builder® and VISION:Two™. Throughout this book, the symbol ⁴ is used to designate that the feature or function being described only applies to VISION:Builder 4000 model series, and not VISION:Two.

Advantage VISION:Builder (hereafter referred to as VISION:Builder) is a software system that can be used to design, implement, and execute data processing applications. Novice users can be productive within a few hours, yet VISION:Builder produces complex applications efficiently. An experienced user can build an application in VISION:Builder in less than half the time it would take for a similar application in COBOL or other third-generation languages.

The philosophy in using VISION:Builder is to:

- Conceptualize how VISION:Builder functions during input, processing, and output
- Design applications that use as many of the automatic functions of VISION:Builder as possible

VISION:Builder provides extensive facilities for the creation and maintenance of files and databases, as well as report generation. VISION:Builder processes virtually any data source and database format available on an IBM® host (for example, flat files, IMS, and DB2) without requiring the programmer to be familiar with these formats.

The VISION:Builder system is made up of the following main components:

- VISION:Builder Engine
- VISION:Workbench for DOS
- VISION:Workbench for ISPF
- COMLIB Library Utilities.

See the [*VISION:Builder Reference Guide*](#) for more information about these components.

Features

This section describes important features and benefits of VISION:Builder.

Efficient Database Access

With VISION:Builder, you can generate virtually an unlimited number of reports in a single pass of multiple databases. Other products require a separate pass of each database for each report, which significantly increase the read/write cycles and runtime.

VISION:Builder is engineered to access all IBM enterprise server databases and files automatically, simultaneously, and concurrently for maximum efficiency.

VISION:Builder frees application developers from the complexities of various database management systems, providing a logical view of data separate from the physical organization. Users preparing their applications are not required to code any calls to DB2[®], IMS/DB, or VSAM database management systems. Once data definitions are prepared and referenced in the program, VISION:Builder automatically manages all the input and output operations for the various databases. This gives you the staffing flexibility you need for application management projects.

Wide Variety of Report Output Formats

With VISION:Builder, you can view reports in a wide variety of formats. In addition to the traditional report formats, VISION:Builder automatically outputs report contents in HTML, TXT, or CSV formats.

Migrating Data

VISION:Builder can select data from existing operational data files and external data sources, and transform, summarize and enhance the data prior to loading into a data warehouse. VISION:Builder can build the initial warehouse and be used on a continuing basis for refreshing the warehouse with updated data.

Meta Data Repository System

The meta data repository system (COMLIB Library Utilities) lets you create file definitions, table definitions, and transaction update definitions once, and then store them in the VISION:Builder library where they can be used for populating and updating data warehouses or by any other program. The library lets you store and reuse VISION:Builder code, so that commonly used procedures can be employed over and over again in different applications.

Verifying the Data

VISION:Builder offers complete transaction processing, as well as extensive editing and automated error-checking for file, database, and data warehouse updates. You can dynamically modify transactions before applying them as updates and test or modify the resulting record. VISION:Builder automatically creates an audit trail.

Handling Complex Applications Easily

VISION:Builder is designed for development organizations with the most challenging enterprise server application requirements. Many VISION:Builder applications in use today involve databases with millions of records and tables with millions of rows.

VISION:Builder helps both novice and experienced developers easily solve complex problems by automating more than 2,000 development functions. Powerful programs that meet a wide variety of objectives can be easily and quickly generated using the defaults and automatic functions of VISION:Builder.

A Computer Associates Solution

Computer Associates provides solutions for global access to corporate data. Its products and services satisfy the data access, applications portfolio management, and desktop integration needs of cross-industry users, regardless of business function.

Benefits

VISION:Builder provides you with the following benefits:

- Automates database conversions necessary for populating and maintaining data warehouses.
- Provides for more efficient use of corporate data with concurrent access to multiple databases.
- Simplifies migration by automating database conversions.
- Minimizes data handling, optimizes runtime, and reduces storage overhead by taking advantage of summarized data files.
- Automates more than 2,000 development functions to simplify development and database management for even the most complex applications.
- Reduces enterprise server overhead by making database access more efficient.
- Provides an efficient and easy-to-use utility to organize corporate data for application management projects.

VISION:Builder Requirements

The following hardware and platforms are required for VISION:Builder:

- All IBM and IBM-compatible enterprise servers — OS/390[®], MVS/ESA[™], VSE/SP/ESA, and VM/XA/ESA.
- Supports VSAM with options for supporting IMS, DB2, and ISV database managers concurrently.
- Optional PC and ISPF program development through VISION:Workbench[™].

Environment

This section describes the packaging and environmental considerations for VISION:Builder Release 14.0.

IBM Operating Systems Supported

Separate versions of VISION:Builder exist to support the various IBM operating systems in use at an installation. The OS/390 version supports all OS/390 and z/OS[™] operating systems. The VSE version supports VSE/ESA[™]. The CMS version supports VM/ESA[®].

Available Interfaces

You can use the following interfaces with VISION:Builder:

- VISION:Workbench for DOS
- VISION:Workbench for ISPF
- DB2 SQL/DS Interface
- VISION:Builder IMS Interface
- Generalized Data Base Interface
- CALL other programs

SMP/E Installation Specifications

The packaging of VISION:Builder Release 14.0 has been changed to conform to the IBM SMP/E standards. SMP/E modification control statements (MCS) along with the supporting JCL for the RECEIVE, APPLY, and ACCEPT processes have been developed to install the elements of the product. The Indirect File method is used to reference the unloaded product data sets during the install process. A REXX program has been written to assist you in tailoring the SMP/E install job streams

(JCL and control statements) to conform to the local shop's standards and conventions. Local product tailoring procedures (M4PARAMS, and so on) are still performed in a manner similar to that used in previous releases of VISION:Builder. However, problem fixes or post-delivery enhancements to the product are now packaged to conform to SMP/E standards and processes for PTFs and/or APARs.

The Function Modifier Identification (FMID) for VISION:Builder Release 14.0 is CCVC140. The FMID for VISION:Two Release 14.0 is CCVPE00.

CA LMP Licensing Specifications

To verify proper authorization to use the product at a customer site, VISION:Builder Release 14.0 uses the CA License Management Program (LMP). The VISION:Builder product family is offered as two products in the marketplace, namely VISION:Builder and VISION:Two. VISION:Builder is the full function product, while VISION:Two is a subset of VISION:Builder that restricts the ability to create new databases and to update existing databases using transaction files.

In addition to the two products noted above, both VISION:Builder and VISION:Two provide the following three database interfaces as optional features:

- DB2 Database Interface
- IMS Database Interface
- Generalized Database Interface

Thus, each customer will use from one to four product codes depending on their licensed features. There will be one product code for the base system, VISION:Builder or VISION:Two, and one code for each licensed optional feature.

The assigned product codes are as follows:

SM	VISION:Builder
SZ	VISION:Builder Interface for GDBI
S4	VISION:Builder Interface for IMS
S6	VISION:Builder Interface for DB2
S8	VISION:Two
S9	VISION:Two Interface for GDBI
TF	VISION:Two Interface for IMS
TG	VISION:Two Interface for DB2

VISION:Builder Release 14.0 has been modified to invoke LMP to verify the authorized use of either the VISION:Builder or VISION:Two products. This verification occurs during product initialization. If the CAIRIM service is not operational, the product terminates with an appropriate error message.

Additionally, VISION:Builder Release 14.0 invokes LMP to verify the authorized use of the base product or any of the three optional database interfaces. If an attempt is made to employ the base product or any of the optional interfaces within an application without proper authorization, an appropriate error message is issued but execution of the product continues.

Documentation

The documentation is delivered on a compact disc (CD). The books are in Adobe Acrobat Portable Document Format (PDF) and are designed for you to read online using the Acrobat Reader (also included on the CD).

Each online book contains a table of contents, index, and underlined colored hypertext links. To go directly to the book, chapter, section, or topic being referenced, click the hypertext link.

The following books are available for VISION:Builder / VISION:Two for OS/390 Release 14.0 on the documentation compact disc:

Name	Description
<i>VISION:Builder for OS/390 Getting Started Guide</i>	<p>Contains fundamental descriptions of VISION:Builder and how to use it, along with a list of enhancements.</p> <p>Lists common applications and tells you where to go for detailed information. Provides you with a road map to the documentation.</p> <p>Contains information specific to the use and operation of the COBOL, DB2®, VISION:Inquiry, and VISION:Results Quick Start utility programs that generate skeletal VISION:Builder file definitions from COBOL data definitions, DB2 catalog information, and VISION:Inquiry and VISION:Results data definitions, respectively.</p>
<u>VISION:Builder for OS/390 Installation and Support Guide</u>	The environment-specific Installation and Support Guide describes how to install VISION:Builder.
<u>VISION:Builder Reference Guide</u>	<p>Describes all of the concepts necessary to understand how VISION:Builder works and how applications can be developed using VISION:Builder. It covers both basic and advanced applications.</p> <p>Also contains information specific to the use of and operation of the toolkit routines delivered with VISION:Builder. These toolkit routines may be used within a VISION:Builder application to perform functions not readily available using native VISION:Builder programming operations.</p> <p>Also provides information specific to the use and operation of a routine which allows user programs written in other languages to access tables stored in a Common Library.</p>
<u>VISION:Builder Reference Summary</u> (previously named <i>VISION:Builder Specifications Manual</i>)	Contains all valid fixed format syntax specifications for VISION:Builder statements.

Name	Description
<i>VISION:Builder ASL Reference Guide</i>	<p>Contains information specific to the use and operation of ASL. (This book assumes that you are familiar with VISION:Builder.)</p> <p>Also describes the conversion from fixed form syntax to Advanced Syntax Language (ASL).</p>
<i>VISION:Workbench for DOS Reference Guide</i>	<p>Contains information specific to the operation and use of VISION:Workbench for DOS. It is designed to give you a basic understanding of the features, capabilities, and flow of VISION:Workbench for DOS.</p>
<i>VISION:Workbench for ISPF Reference Guide</i>	<p>Contains a brief introduction to VISION:Workbench for ISPF, an illustrated application, and an in-depth discussion of the panels in VISION:Workbench for ISPF.</p>
<i>VISION:Builder for OS/390 Environment Guide</i>	<p>Describes, in separate books, detailed information for the use of VISION:Builder and the COMLIB system under the OS/390 and z/OS environment.</p>
<i>VISION:Builder Messages and Codes</i>	<p>Lists VISION:Builder messages and codes and gives detailed explanations.</p> <p>Also assists programmers in identifying and resolving problems that may occur during the execution of VISION:Builder.</p>

The following books used to be delivered with VISION:Builder and are now incorporated into other books:

Old Name	Description	Information Now Contained in this Book
<i>VISION:Builder Toolkit Reference Manual</i>	<p>Contains information specific to the use of and operation of the toolkit routines delivered with VISION:Builder. These toolkit routines may be used within a VISION:Builder application to perform functions not readily available using native VISION:Builder programming operations.</p> <p>Also provides information specific to the use and operation of a routine which allows user programs written in other languages to access tables stored in a Common Library.</p>	<i>VISION:Builder Reference Guide</i>
<i>VISION:Builder New Release Planning Guide</i>	<p>Contains the general and technical information about the new releases of VISION:Builder and VISION:Workbench for DOS.</p> <p>This document complements other VISION:Builder and VISION:Workbench for DOS manuals by providing a summary of the changes and enhancements for the new releases and information pertinent to migrating to the new releases.</p>	<i>VISION:Builder for OS/390 Getting Started Guide</i>
<i>VISION:Builder Specifications Manual</i>	Contains all valid fixed format syntax specifications for VISION:Builder statements.	<i>VISION:Builder Reference Summary</i>
<i>VISION:Builder Conversion Manual</i>	Describes the conversion from fixed form syntax to ASL and VISION:Workbench for DOS window entries.	<i>VISION:Builder ASL Reference Guide</i>

Old Name	Description	Information Now Contained in this Book
<i>VISION:Builder Problem Determination Guide</i>	Assists programmers in identifying and resolving problems that may occur during the execution of VISION:Builder.	<u>VISION:Builder Messages and Codes</u>
<i>COMLIB System Messages</i>	Lists message text and detailed explanations for COMLIB system messages.	<u>VISION:Builder Messages and Codes</u>

The following books only apply to customers who have the GDBI feature. These books are available only by special request and are not part of the CDROM documentation kit:

Name	Description
GDBI Programming Guide	Covers the concepts and operation of the Generalized Data Base Interface which enables you to access and process data from virtually any Database Management System without having to be concerned about where the data is stored or in what format it is stored.
Using GDBI with ADABAS	Aids you in the design and implementation of VISION:Builder applications which access Software AG's Adaptable Data Base System (ADABAS) files using the GDBI option of VISION:Builder.
Using GDBI with SUPRA	Aids you in the design and implementation of VISION:Builder applications which access Cincom's SUPRA databases using the GDBI option of VISION:Builder.
Using GDBI with IDMS	Aids you in the design and implementation of VISION:Builder applications which access Computer Associate's Integrated Database Management System (IDMS) databases using the GDBI option of VISION:Builder.

Installing Online Documentation and the Acrobat Reader

You can install the online documentation on your local hard drive or on a network server. Or, you can access the documentation directly from the compact disc.

If you do not have Acrobat Reader installed, you can install it from the compact disc.

To install the online documentation, the Acrobat Reader, or both:

1. Close all application programs, including screen-saver software.
2. Insert the compact disc into the CD-ROM drive.
3. Click the Start menu and select Run.
4. In the Run dialog box, type: D:\Books\Setup.exe (where D: is the CD-ROM drive) and click OK.
5. You can select the option to install the online documentation or the Acrobat Reader.
6. Follow the instructions.

Viewing Online Documentation

Regardless of the location of the online documentation (on a hard drive or compact disc), you can view the online documentation using the following methods:

- Point to the directory on the hard drive or CD-ROM drive and double-click the PDF file.
- Click the Start menu, point to Programs, point to Advantage VISION_Builder VISION_Two 14.0 OS, and click the document title.

Educational and Professional Services

You can become proficient in using VISION:Builder by reading the [*VISION:Builder Reference Guide*](#) and writing several programs. However, if you would like more extensive training, Computer Associates offers two courses designed to assist you in getting the most from your investment in VISION:Builder:

- **VISION:Builder Basic Concepts**

This five-day course, which includes hands-on workshops, provides an introduction to the basic features and capabilities of VISION:Builder. From simple applications to more sophisticated techniques, this course provides an in-depth understanding of the VISION:Builder language and special features.

- **VISION:Builder Advanced Concepts**

This five-day course provides an introduction to VISION:Builder's advanced features and capabilities. Simple to advanced applications are covered in each area of interest. You can add DB2 and IMS[™] interfaces by request.

Classes are held at various regional locations, or you can make arrangements to have the class taught at your site.

In addition to educational services, Computer Associates offers consulting and programming services to assist you in exploiting the full capabilities and power of VISION:Builder. See [Contacting Computer Associates](#) for information about contacting Computer Associates Professional Services.

Contacting Total License Care (TLC)

TLC is available Monday-Friday 7 AM - 9 PM Eastern time in North America and 7 AM - 7 PM United Kingdom time. Additionally, 24-hour callback service is available for after hours support. Contact TLC for all your licensing requirements.

Be prepared to provide your site ID for product activation.

To activate your product (ask for a license key or validation), use one of the following:

Location	Phone	Email
North America	800-388-6720 (toll free) 631-342-5069	help@licensedesk.cai.com
Europe	00800-1050-1050	euro.tlc@ca.com
If your company or local phone service does not provide international access, please call your local Computer Associates office and have them route you to the above number.		
Australia:	1-800-224-852	
New Zealand:	0-800-224-852	
Asia Pacific:	800-224-852	
Brazil:	55-11-5503-6100	
Japan:	Not available	JPNTLC@ca.com

Contacting Computer Associates

For technical assistance with this product, contact Computer Associates Technical Support on the Internet at esupport.ca.com. Technical support is available 24 hours a day, 7 days a week.

Enhancements and Modifications

This chapter describes the changes for VISION:Builder Release 14.0.

In VISION:Builder Release 14.0, the implementation of the ASL (Advanced Syntax Language) feature has been completed so that you can now code an entire application in ASL. Therefore, you no longer need to code portions of your application using the traditional fixed-form-syntax language specifications. Previously, ASL only supported the procedural portion of the VISION:Builder language specifications. New ASL statements are now provided for Run Control, Reporting, Extracted Data Files (Subfiles), and the small subset of Procedure statements that previously required the use of fixed-form-syntax.

Both the existing ASL statements and all new ASL statements now allow the use of long field names (up to 30 characters) in VISION:Builder Release 14.0. Previously, ASL statements as well as the traditional fixed-form-syntax specifications restricted field names to a maximum of eight characters. Fixed-form-syntax statements continue to restrict field names to a maximum of eight characters.

In addition to the ASL extensions, the previous restriction of only allowing a maximum of 10 extracted data files to be created within a single application has been lifted in VISION:Builder Release 14.0. This enhancement significantly improves the power of VISION:Builder for data extraction tasks. Because each “extracted data file” can be used to directly populate a DB2 table, this enhancement significantly strengthens the power of VISION:Builder for data warehouse construction applications.

This section describes the enhancements for VISION:Builder Release 14.0.

The following list summarizes the major functional enhancements in VISION:Builder Release 14.0:

- Extend ASL to support the Run Control statement specifications
 - SQL Clause on FILE statements
 - SEGMENT and WHERE clause on FILE statements
 - SEGMENT and SSA clause on FILE statements
- Extend ASL to support Report statement specifications

-
- Extend ASL to support the EXTRACT statement in procedures
 - Extend ASL for procedures to eliminate the need for any fixed-form-syntax statements
 - Long field names (up to 30 characters)
 - Remove the current restriction of only 10 extracted data files (subfiles) per application (allow up to 138 extracted pre-sort data files)
 - Allow extracted data files to be produced following the sort
 - Conform to the IBM SMP/E product packaging conventions
 - Conform to the CA LMP license verification process

The option for developing a fully ASL coded application in VISION:Builder Release 14.0 makes it easier for you to assimilate, understand, and prepare applications. The following sections provide the detail functional specifications for each of the major new functional enhancements.

As in previous releases of VISION:Builder, use the following conventions to specify the syntax of each command or function:

- Brackets [] indicate an optional keyword or parameter.
- Braces { } indicate a choice of entry. Unless a default parameter is indicated by an underscored entry, you must choose one of the entries.
- Required parameters do not have brackets or braces surrounding them.
- Items separated by a vertical bar (|) represent alternative items. Select only one of the items.
- Items separated by a plus sign (+) represent alternative items. Select as many as appropriate.
- An ellipsis (...) indicates that you can use a progressive sequence of the type immediately preceding the ellipsis. For example: name1, name2, ...
- Bold uppercase type indicates the characters to be entered. Enter such items exactly as illustrated. You can use an abbreviation if it is indicated.
- Italic lowercase type indicates parameters to be supplied by the user.
- Underscored type indicates a default option. If you omit the parameter, the underscored value is assumed.
- Separate commands, keywords, and operands by blanks.
- Enter punctuation exactly as shown (parentheses, colons, and so on).
- Continue a command on a new line by ending the line with a comma.
- Comments are entered following a semi-colon.

General Enhancements

ASL Run Control Statements

Previously, all Run Control statements of VISION:Builder (those specifying the basic control parameters and file usage for a VISION:Builder application) had to be coded in the traditional fixed-form-syntax. VISION:Builder Release 14.0 now accepts new ASL statements that you can use to specify basic control parameters and the file usage for an application. The Run Control statements and their syntax are as follows:

```
CONTROL [NAME run-name],
        [DELIMITER 'x'],
        [{SCANONLY | SAMPLE | MAPDECODE}],
        [{TERM | CONTINUE}],
        [SORT {INTERNAL | EXTERNAL | NONE}, [SUMMARIZE],
        [{NOLIST | NOSOURCE | LISTGEN}],
        [[AMODE {31 | 24}],
        [ABEND],
        [GRANDSUM],
        [CORDONLY],
        [GETMAIN nnnK],
        [SORTSIZE nnnK],
        [REPTSIZE nnnK],
        [FREESIZE nnnK],
        [DB2 sub-system-id plan-name],
        [SQLID sql-authorization-id],
        [EXPLAIN nnn],
        [{SYSDATE mmddyy | SYSDATE4 yyyyymmdd}],
        [DECMGS {YES | NO}],
        [PROMSGS {YES | NO}],
        [RPTMSGs {YES | NO}]

FILE MASTER {INPUT | UPDATE_IN_PLACE},
        [{NAME definition-name | SQL "sql-select-statement"}],
        [ACCESS {SEQUENTIAL | DIRECT | PHYSICAL}],
        [KEYS {UNIQUE | EQUAL | NONE}],
        [STARTKEY key-value],
        [ENDKEY key-value],
        [KEYNAME field-name ...],
        [MOSAIC],
        [CHKORDER],
        [ONE_BUFFER],
        [{SEGMENT segment-name WHERE "sql-where-clause" ... |
        SEGMENT segment-name SSA "pre-selection-ssa" ... }]

FILE MASTER OUTPUT,
        [NAME] definition-name

FILE MASTER DUMMY,
        [NAME] definition-name
```

The following rules apply to FILE MASTER statements:

- Both FILE MASTER INPUT and FILE MASTER OUTPUT may be present in the same application.
- FILE MASTER OUTPUT is not allowed when FILE MASTER UPDATE is present.

- FILE MASTER DUMMY is not allowed when any other FILE MASTER statements are present.
- When FILE MASTER DUMMY is present, FILE TRAN (see below) must also be present.

```
FILE CORDn [{NAME definition-name | SQL "sql-select-statement"}] (n = 1 to 9)
    [MOSAIC],
    [CHKORDER],
    [ARRAY],
    [{SEGMENT segment-name WHERE "sql-where-clause" ... |
     SEGMENT segment-name SSA "pre-selection-ssa" ... }]
    [DIRECT BY q.fldname] (indexed direct - ICF)
    [STANDARD], (standard coordination)
    [{ALLRECS | MATCHONLY}],
    [KEYNAME field-name ...]
    [CHAIN TO qual], (chained coordination)
    [KEYNAME field-name ...]
    [USER READ], (user read)
    [GENERIC field-name]
```

```
FILE TRAN [NAME definition-name],
    [GROUPS group-name ...],
    [CHKORDER]
```

FILE REJECT

FILE AUDIT

```
FILE SUBFn [NAME] subfile-name, (n = 0 to 9)
    [TABLE authid.tablename {CREATE | DELETE | INSERT | DROP}],
    [TABLESPACE tablespace-name],
    [DATABASE database-name],
    [DROP_DELETED_SEGMENTS],
    [AUTODEF]
```

FILE REPORT

```
FILE REPn [NAME] report-file-name (n = 2 to 9)
```

The following additional keywords are common to all of the above FILE statements except FILE MASTER DUMMY:

```
[{PASSWORD password | AUTHID authorization-id}]
[IO_PLUGIN module-name]
[DDNAME ddname]
```



```

ARRAY [QUALIFIER] qual-char,
      NAME definition-name,
      [OVER_DEFINES qual-char]

WORK [AREA] number,                                (number = 01 to 99)
     NAME definition-name

LINKAGE [AREA] number,                              (number = 01 to 99)
      NAME definition-name

ROUTE {[REPORTS] report-name ... | ALL},
      [KEYVALUE 'data-value'],
      TO destination-names ...,
      [DEFER]

COLLATE {[REPORTS] report-name ... [KEYLENGTH length] |
        ALL KEYLENGTH length}

CHECKPOINT [FILE ddname],
          [COUNT number],
          [TIME number {MINUTES | SECONDS}],
          [EOV ddname],
          [OPERATOR],
          [PREFIX id-prefix],
          [COMMITONLY],
          [ALTERNATING]

CATALOG {SAVE [GROUP group-name] [REQUEST request-name ...] |
        INSERT REQUEST request-name INTO group-name [AFTER request-name] |
        DELETE [GROUP group-name] [REQUEST request-name ...] |
        REPLACE REQUEST request-name ... |
        DUMP {ALL | ITEMS name ... } |
        LIST}

TRACK [NAME] item-name,
      [GENERIC],
      [TYPE {FILE | ARRAY | TABLE | TRAN | REQUEST | REQGROUP}],
      [FILENAME file-name],
      [{EXPIRE mmddyy | RETAIN days}],
      [USERID userid]

OVERRIDE [DDNAME] old-ddname WITH new-ddname

MULTILIB {OFF | ORDER m4libn ...}

OWNCODE [MODULE] module-name,
        HOOKS hook-number ...

LISTCNTL [ALT_LIST {YES | NO}],
        [FILESUM {YES | NO}],
        [INDEF {YES | NO}],
        [INGLOSS {YES | NO}],
        [INREQ {YES | NO}],
        [CATREQ {YES | NO}],
        [MAPREQ {YES | NO}],
        [SQLSTAT {YES | NO}],
        [MOSAICSTAT {YES | NO}]

DOCUMENT [CONVMGS],
        [XREF],
        [EXECTRACE],
        [MAXLINES nnnK]

LISTLIB NAMES {ALL | ARRAY | FILE | GROUP | TABLE | VIEW}

LISTLIB GLOSSARY {ARRAY | FILE | GROUP | TABLE | VIEW | WITHIN_VIEW},
        {ALL | ITEMS name ...}

RETRIEVE {ARRAY | FILE | GROUP | REQUEST | TABLE | VIEW | EOF},

```

```
{ALL | ITEMS name ...},  
[NEWNAME name]
```

No FILE statements may be present when you use the LISTLIB or RETRIEVE command.

```
DEBUG [CLEAR],  
      [DUMP],  
      [EXIT],  
      [LONGNAMES],
```

(internal debugging controls)

```
COPY [DDNAME] ddname[(member-name)] [FIXED]
```

You can include the COPY statement anywhere in the run-control section. The statements in the referenced file are copied into the source code input stream and processed in an identical manner as if the statements had been in-stream.

If you specify the FIXED keyword on the COPY statement, the statements are assumed to be VISION:Builder fixed-form-syntax statements and are processed as such. Copied members from a COPY without the FIXED keyword may in turn contain COPY statements.

A COPY statement with the FIXED keyword signals the end of the Run Control section of the application. This means that none of the above statements may appear following a COPY statement with the FIXED keyword.

Typically, a COPY statement with the FIXED keyword would either include definitions from a Definition Library (created using the Workbench for ISPF) or fixed-form-syntax statements for requests.

SQL Clause on FILE Statement

In VISION:Builder Release 14.0, if the MASTER or CORDn file is a DB2 file, VISION:Builder accepts an SQL clause on the FILE statements for these files. The SQL clause must contain a complete DB2 SELECT statement that is used to access the DB2 table rows for the related file. When the DB2 clause is present, no pre-existing file definition for the related file is needed. VISION:Builder constructs a file-definition automatically on-the-fly from the information obtained from the DB2 catalog for the SELECT statement.

There are no restrictions on the scope of the SELECT clause included with the FILE statement; any legal DB2 SELECT statement is allowed. You cannot use the following FILE statement keywords when the SQL clause is present: OUTPUT, STARTKEY, ENDKEY, SEGMENT, and WHERE. You can use the KEYNAME parameter on the FILE statement to identify the key field(s) in those cases where the DB2 Catalog does identify the key columns in the table. The following is an example of a VISION:Builder application using the SQL clause:

```
; Sample Builder Application Reading Data using an SQL SELECT Statement
;
CONTROL DB2 D61A INM4CALL
;
FILE MASTER INPUT, KEYNAME A.DEPT_NUMBER B.EMP_NUMBER,
      SQL "SELECT A.DEPT_NUMBER, A.DEPT_NAME, B.EMP_NUMBER ",
          "B.LAST_NAME, B.FIRST_NAME, B.MIDDLE_INITIAL ",
          "FROM DSN8610.DEPT A, DSN8610.EMP B ",
          "WHERE A.DEPT_NUMBER = B.WORK_DEPT ",
          "ORDER BY A.DEPT_NUMBER, B.EMP_NUMBER"
;
FILE REPORT          ;Indicates that a Report File is needed
;
MAIN: PROC
      REPORT DEPT_NUMBER, DEPT_NAME, EMP_NUMBER, LASTNAME,
            FIRSTNAME, MIDDLE_INITIAL
            TITLE 'EMPLOYEE LIST BY DEPARTMENT'
            GROUP BY DEPT_NUMBER DEPT_NAME, SUBTITLE
            COUNT EMP_NUMBER BY DEPT_NUMBER
      END REPORT
END PROC
;
; End of Sample Application
```

SEGMENT and WHERE Clause on FILE Statement

Optionally, you can use the SEGMENT and WHERE clause on a FILE MASTER INPUT, FILE MASTER UPDATE, or FILE CORDn statement to qualify the SQL SELECT statement generated by VISION:Builder for a DB2 database. This lets you qualify the table rows that will be retrieved and optimize the access to DB2 databases. The function of the SEGMENT and WHERE clause is identical to the fixed-form-syntax WH statements.

The following is an example of SEGMENT and WHERE clause usage for a DB2 database:

```
FILE MASTER INPUT, NAME DEPT EMP,
      SEGMENT EMPLOYEE, WHERE "BIRTH_DATE < '1970-01-01' AND SALARY > 50000"
```

Use of the SEGMENT and WHERE clause on a FILE statement is mutually exclusive with the use of the SQL clause.

SEGMENT and SSA Clause on FILE Statement

You can optionally use the SEGMENT and SSA clause on the FILE MASTER INPUT, FILE MASTER UPDATE, or FILE CORDn statement to qualify the DL/I GetUnique or GetNext calls for segments of an IMS database. When VISION:Builder constructs the required parameter list to perform the appropriate DL/I Get calls, any SSA clause qualifications are appended to the parameters. This

lets you qualify the IMS database segments that will be retrieved and optimize the number of DL/I calls required. The function of the SEGMENT and SSA clause is identical to the fixed-form-syntax Pre-Selection Requests.

The following is an example of a SEGMENT and SSA clause usage for an IMS database:

```
FILE MASTER INPUT, NAME CUSTOMER,  
  SEGMENT CUSTOMER, SSA "STATE = 'CA' OR STATE = 'AZ'",  
  SEGMENT ORDER,    SSA "AMOUNT > 1000"
```

ASL Report Statements

Additional ASL statements are provided to allow the specifications of reports. The REPORT statement, along with its subordinate statements, is known as a Report Block. You can include as many Report Blocks as you want within an ASL procedure. The first statement of a Report Block must be a REPORT statement with an optional label specifying the name of the Report Block. The label on the REPORT statements is required only when you specify ROUTE and COLLATE statements for the report. A Report Block ends when an "END REPORT" statement is encountered. The Report Block statements and their syntax are as follows:

```
report-name: REPORT [COLUMNS] {q.fldname | report-function} ...,  
                  [SUMMARY ONLY],  
                  [GRANDSUMS],  
                  [EMPTY FIELD {INCLUDE | EXCLUDE}],  
                  [SELECTION CONTROL {YES | NO}],  
                  [MAXITEMS number],  
                  [ABBREVIATED],  
                  [FILE report-file-name]  
                  {TYPE {NORMAL | SUBROUTINE | INIT | PRE MASTER READ |  
                        TYPE1 | TYPE2 | TYPEM | TYPE3 | TYPE4 |  
                        EOF | EOFPLUS}},  
                  [INFO 'text']  
  
TITLE [LINE] 'text' ...  
  
ORDER [BY] q.fldname ... [DESC q.fldname ...]  
  
GROUP [AT LEVEL number] [BY] q.fldname ...  
      [SUBTITLE [NEWPAGE]],  
      [LABEL]  
  
FORMAT [HEIGHT number],  
       [WIDTH number],  
       [DATEPOS {UL | UR | UM | LL | LR | LM | NO}],  
       [PAGEPOS {UL | UR | UM | LL | LR | LM | NO}],  
       [TITLEPOS {TOP | BOTTOM}],  
       [LABELS {SPACE | NOSPACE | SUPPRESS}],  
       [HEADINGS {YES | NO | NAME}],  
       [HEADPOS {ABOVE | BELOW}],  
       [DATEFMT {DATE | TODAY | TODAYX | ISDATE | JULIAN | JULIANX |  
                mmdyy}],  
       [LINES number],  
       [BORDER [YES | NO | 'x']],  
       [STARTPAGE {number | PAGE}],  
       [MAXPAGES number],  
       [LINESPERPAGE number],  
       [DETAILSPACING number],  
       [INCOMPLETESUM char],  
       [NODATA {SKELETON | NOREPORT}],  
       [SUBTITLE {REPEAT | NOREPEAT | NEWPAGE}],  
       [SUMMARYLABELS {SPACE | NOSPACE | SUPPRESS}],  
       [LINENUMS {NONE | LEFT | RIGHT | BOTH}],  
       [IMAGES number] [IMGTITLE {LOGPAGE | PHYPAGE | NEWPAGE}]
```

```

[METHOD {STD_LIST | ALT_LIST | CSV | TAB | HTML | PLAIN_TEXT |
          RAW_DATA }],
[STYLE number],
[DDNAME ddname],
[SUMFILE ddname],
[AUTODEF]

ITEM [COLUMN] q.flname ...
      [SPACES number],
      [PICTURE P'pattern'],
      [ENDLINE],
      [NONPRINT],
      [VWIDTH],
      [NOWRAP],
      [SPLITOK],
      [CSVEDIT {QUOTE | TRUNCATE [DECIMALS number]}]

TOTAL [ITEM] q.flname ... {BY q.flname | AT LEVEL number}
CUM [ITEM] q.flname ... {BY q.flname | AT LEVEL number}
COUNT [ITEM] q.flname ... {BY q.flname | AT LEVEL number}
MAX [ITEM] q.flname ... {BY q.flname | AT LEVEL number}
MIN [ITEM] q.flname ... {BY q.flname | AT LEVEL number}
AVG [ITEM] q.flname ... {BY q.flname | AT LEVEL number}
PCT [ITEM] q.flname ... OF q.flname {BY q.flname | AT LEVEL number}
RATIO [ITEM] q.flname ... TO q.flname {BY q.flname | AT LEVEL number}
PREFACE [LINE] 'text' ...
XREP [LINE] 'text' ...      (text may include any MARKXREP control commands)
END [REPORT]

```

Built-in Report Functions

In addition to the above statements, six built-in functions are provided for use in the REPORT statement to simplify the coding required. You can specify the following built-in functions as a COLUMNS operand of the REPORT statement in place of a simple qualified name:

```

PF([FIELD] q.flname [START] start-position [[LENGTH] partial-length])
TOTAL([ITEM] q.flname BY q.flname)
COUNT([ITEM] q.flname BY q.flname)
MAX([ITEM] q.flname BY q.flname)
MIN([ITEM] q.flname BY q.flname)
AVG([ITEM] q.flname BY q.flname)

```

When you use these built-in functions, a column for the field-name is included in the report. If the function is a summary function (TOTAL, COUNT, MAX, MIN, AVG), the summary for that field is included, as well. When you use any of these functions (except for the PF function) as a COLUMNS operand on the REPORT statement, the explicit TOTAL, COUNT, MAX, MIN, or AVG statements described earlier are not necessary.

Sectional Reporting Extensions

You can use the following statements within a REPORT block to specify section layout and content for the page-title, column-heading, or summary sections of a report. The specifications for a section must begin with a SECTION statement and end with an END statement. You can include only one specification for each section type within a REPORT block. If no SECTION statements are provided, the default layout and content is used for the report.

SECTION PAGE_TITLE or SECTION COLUMN_HEADING

LINE {'text' | LITERAL('text', num-times) | q.fldname | COL(col-num) |
SPACES(num-spaces)}...

SKIP num-lines LINES

END [SECTION]

SECTION SUMMARY

COMPUTE STEMPnn = operand operator operand [PIC P'picture'] [DECIMALS number],
[ROUNDED] AT LEVEL level

DATA {'text' | LITERAL('text', num-times) | q.fldname | sumftn(q.fldname) |
STEMPnn |
COL(col-num) | SPACES(num-spaces)}... AT LEVEL level

LINE [{ 'text' | LIT[ERAL]('text', num-times) | q.fldname | sumftn(q.fldname) |
STEMPnn,
COL(col-num) | SPACES(num-spaces)}...] [AT LEVEL level]

SKIP num-lines LINES [AT LEVEL level]

END [SECTION]

Semantics:

- Level = 1 to 9, or G[RAND] except for DATA statement where level = 0 to 9 or G[RAND]
- DATA statement(s) must be followed by a LINE statement; multiple DATA statements allowed prior to a LINE statement
- sumftn = TOT[AL], CUM, COUNT, MAX, MIN, AVG, PCT, RAT[IO]

General Report Semantics

The Report Block statements are translated to the appropriate ER, En, Rn, Pn, Tn, and Fn fixed-form-syntax statements.

COLUMNS on the REPORT statement may be augmented by columns from the ORDER, GROUP, ITEM, TOTAL, CUM, COUNT, MAX, MIN, AVG, PCT, and RATIO statements. In other words, if a column name appears both on the REPORT statement and on any of the other statements, the parameters are combined to form the complete specification for that entry on the generated Rn statement. If a column is referenced only on an ORDER or GROUP statement, the field is considered as NONPRINT. You can use an ITEM statement for the same field without the NONPRINT keyword to override this default NONPRINT designation.

ASL EXTRACT Statement

A new ASL EXTRACT statement is provided to output an extracted data file (subfile). This statement must be contained within an ASL procedure. There are four variations of the EXTRACT statement. The first variation requires a corresponding "FILE SUBFn" statement, while the remaining ones do not.

The syntax of the EXTRACT statement requiring a corresponding "FILE SUBFn" statement is:

```
EXTRACT FILE subfile-name ,
    {[ITEMS] {q.fldname | pf-function} ...} | ENTIRE qual-char},
    [KEYS q.fldname ...],
    [VARCHARMAX NOPREFIX q.fldname ...],
    [VARCHARMAX WITHPREFIX q.fldname ...],
    [SELECTION_CONTROL {YES | NO}],
    [MAXITEMS number],
    [ABBREVIATED],
    [RECFM {FIXED | VARIABLE | UNDEFINED | KEY_VSAM | ENTRY_VSAM |
        DLI | HDAM | DB2 | PACKED}],
    [BLKSIZE number],
    [AUTODEF],
    [TYPE {NORMAL | SUBROUTINE | INIT | PRE MASTER_READ |
        TYPE1 | TYPE2 | TYPEM | TYPE3 | TYPE4 |
        EOF | EOFPLUS}],
    [INFO 'text']
```

This syntax of the EXTRACT statement used to create a subfile as either a sequential file or a VSAM file without a corresponding "FILE SUBFn" statement is:

```
EXTRACT DDNAME ddname,
    {[ITEMS] {q.fldname | pf-function} ...} | ENTIRE qual-char},
    [KEYS q.fldname ...],
    [VARCHARMAX NOPREFIX q.fldname ...],
    [VARCHARMAX WITHPREFIX q.fldname ...],
    [SELECTION_CONTROL {YES | NO}],
    [MAXITEMS number],
    [ABBREVIATED],
    [RECFM {FIXED | VARIABLE | UNDEFINED | KEY_VSAM | ENTRY_VSAM |
        PACKED}],
    [BLKSIZE number],
    [AUTODEF],
    [TYPE {NORMAL | SUBROUTINE | INIT | PRE MASTER_READ |
        TYPE1 | TYPE2 | TYPEM | TYPE3 | TYPE4 |
        EOF | EOFPLUS}],
    [INFO 'text']
```

The syntax of the EXTRACT statement used to create a subfile as an IMS database without a corresponding "FILE SUBFn" statement is:

```
EXTRACT DBDNAME dbdname,
    ENTIRE qual-char,
    [{DLI | HDAM}],
    [SELECTION_CONTROL {YES | NO}],
    [MAXITEMS number],
    [AUTODEF],
    [TYPE {NORMAL | SUBROUTINE | INIT | PRE MASTER_READ |
        TYPE1 | TYPE2 | TYPEM | TYPE3 | TYPE4 |
        EOF | EOFPLUS}],
    [INFO 'text']
```

The syntax of the EXTRACT statement used to create a subfile as a DB2 table without a corresponding "FILE SUBFn" statement is:

```
EXTRACT TABLE "authid.tablename",
  {CREATE | DELETE | INSERT | DROP},
  [TABLESPACE tablespace-name],
  [DATABASE database-name],
  ITEMS {q.fldname | pf-function} ... },
  [KEYS q.fldname ...],
  [SELECTION_CONTROL {YES | NO}],
  [MAXITEMS number],
  [ABBREVIATED],
  [AUTODEF],
  [DEFNAME defname],
  [TYPE {NORMAL | SUBROUTINE | INIT | PRE_MASTER_READ |
        TYPE1 | TYPE2 | TYPEM | TYPE3 | TYPE4 |
        EOF | EOFPLUS}],
  [INFO 'text']
```

Additional ASL Procedure Statements

New PROC, INCLUDE, and COPY statements are introduced in the procedure section, allowing you to code a procedure entirely in ASL. The PROC statement eliminates the previous requirement that required a fixed-form-syntax ER statement to precede ASL procedure statements. The PROC statement begins a procedure block that must be terminated by an END statement. The syntax of the PROC statements is as follows:

```
proc-name: PROC [TYPE {NORMAL | SUBROUTINE | INIT | PRE_MASTER_READ |
                     TRAN1 | TRAN2 | TRANM | TRAN3 | TRAN4 |
                     EOF | EOFPLUS}],
  [REINIT TEMPS],
  [SELECTION_CONTROL {YES | NO}],
  [MAXITEMS number],
  [INFO 'text'],
  [PARALLEL_LOOPING]
```

The INCLUDE statement eliminates the previous requirement to code a fixed-form-syntax CR statement to include a cataloged procedure, request, or request group. The object to be "included" must have been previously cataloged into the Common Library. The syntax of the INCLUDE statement is as follows:

```
INCLUDE [ITEM] item-name [DATEFMT {DATE | TODAY | ISDATE | JULIAN | mmdyy}],
  [INFO 'text']
```

You can use the COPY statement to copy additional ASL statements into the procedure section of an application just as if they were included in the input stream. If the copied member contains fixed-form-syntax statements, the COPY statement must not be within the scope of a procedure block and you must specify the FIXED keyword. Otherwise, the copy statement may occur anywhere in the input stream. Copied members from a COPY without the FIXED keyword may in turn contain COPY statements. The syntax of the COPY statement for procedures is as follows:

```
COPY [DDNAME] ddname[(member-name)] [FIXED]
```


Use the END statement to terminate a procedure block that began with a PROC statement.

```
END [PROC]
```

You can use all of the ASL procedural statements available in previous releases of VISION:Builder within a procedure block.

Also, using the fixed-form-syntax ER statement followed by the ##PROC and ##PEND statements to bracket ASL procedural statements will continue to be supported. When you use this convention, you cannot use the PROC, INCLUDE, and COPY statements within the scope of the ##PROC/##PEND group.

Long Field Names

Traditionally, data element names in VISION:Builder language statements are still limited to a maximum of eight characters. This has been, and will continue to be, a restriction imposed by the fixed-form-syntax statements. Beginning with Release 12.0 of VISION:Builder, users could define data elements with both a short (8-character) and a long (30-character) name. The DB2 and COBOL Quick Start utilities could also be used to create data definitions with data element names longer than 8 characters. However, the only place that the long names could be used was in a VISION:Inform query.

In VISION:Builder Release 14.0, statements coded with the ASL syntax can now use data element names that are longer than eight characters. This makes the language statements more readable and eliminates the need to use short alias names for data definitions imported from DB2 and COBOL that contained names longer than eight characters.

Remove Limit of 10 Extracted Data Files (Subfiles)

In previous releases of VISION:Builder, the number of subfiles in an application was limited to 10. In VISION:Builder Release 14.0, 138 subfiles are allowed (the basic 10 subfiles plus an additional 128 extended subfiles.). The previous subfile functionality specified using the "FILE SUBFn file-name ..." statements in combination with the "EXTRACT FILE file-name ..." statements is still supported. In addition, three alternate forms of the EXTRACT statement (namely EXTRACT DDNAME, EXTRACT DBDNAME, and EXTRACT TABLE) are now accepted in VISION:Builder Release 14.0. These alternate forms of the EXTRACT statements do not require a corresponding FILE statement and, as such, are not restricted by the previous limit of 10 "FILE SUBFn file-name ..." statements.

The new fixed-syntax specifications for the En Report Handling entry (cols. 41-42) are as follows:

Entry	Result
DD	A pre-sort extended subfile and a subfile file definition is created. An RF statement with a C in the CORD entry (col. 53) is required when this En statement specification is used.
DO	A pre-sort extended subfile is created.

No RF statement for the subfile is required when the above Report Handling codes are used. The En statement Subfile Format entry (col. 56) determines the type of subfile created and the meaning of the DTF/DDname or File Name entry (cols. 44-51). The additional En statement specifications for the DTF/DDname or File Entry (cols. 44-51) are as follows:

Extended Subfile Usage of File Name Entry (Cols. 44-51)

- Enter a DBDNAME when the Subfile Format entry is D or H.
- Leave blank when the Subfile Format entry is S and the Report Handling Entry is DO.
- Enter a file definition name for the created definition when the Subfile Format entry is S and the Report Handling entry is DD.
- Enter a DDname when the Subfile Format is any code other than D, H, or S.

When a Report Handling entry of DD or DO is used in conjunction with a Subfile Format of S, a new Wn statement is required. The layout of the Wn statement and the specifications rules are identical to the RT statement, with the exception that the File Name entry (cols. 11-18) of the Wn statement is not a required entry and should be left blank.

Extracted Data Files Following the Sort

Previously, extracted data files were only created during the file-processing phase of a VISION:Builder application before any sorting took place. Thus, the extracted files were always in the same sequence as the input data. VISION:Builder Release 14.0 supports the ability to produce extracted data files from the report phase following the sort. This way, the records in the extracted data files are ordered just as report data is ordered. These post-sort extracted data files are specified using the REPORT statement that includes a "FORMAT ... METHOD RAW_DATA ..." statement within the Report Block. Post-sort extracted data file records always use the variable-length record (RECFM V) convention.

The new fixed-syntax specifications for the En Report Handling entry (cols. 41-42) are as follows:

Entry	Result
RD	A post-sort extended subfile and a subfile file definition is created. An RF statement with a C in the CORD entry (col. 53) is required when this En statement specification is used.
RO	A post-sort extended subfile is created.

No RF statement for the subfile is required when the above Report Handling codes are used. The En statement Subfile Format entry (col. 56) must be left blank, and the DTF/DDname or File Entry (cols. 44-51) must contain a DDname to which the file will be written. This will always output as variable length record file (RECFM V).

Customer-Requested Enhancements

The following enhancements have been implemented in VISION:Builder Release 14.0 in response to customer initiated requests. These enhancements include the requests identified by DARs 10605358, 10605928, 10606035, and 10606090.

PL/I-Like Varchar Output

Previously, V-type fields output through a subfile were always formatted according to the VISION:Builder conventions. Because this format was not compatible with PL/I varying length character field formats, the only application that could read these subfiles was VISION:Builder itself. This enhancement allows the application developer to specify that VISION:Builder is to output V-type fields as either PL/I-like varchar fields or as long character fields so that non-VISION:Builder applications can read the files.

The new fixed-syntax specifications for the Rn statement Control entry (col. 29) are as follows:

Entry	Result
X	The V-type field will be output as a PL/I compatible varchar field. That is, the field will be output with a 2-byte prefix containing the current length of the field followed by the significant characters in the field (represented by the current length) padded with blanks up to the maximum length of the field.
Y	The V-type field will be output as a long character field containing the significant characters in the field (represented by the current length) padded with blanks up to the maximum length of the field.

The ASL examples below are the equivalent way of specifying the fixed-syntax specifications:

```
EXTRACT ITEMS ...,
      VARCHARMAX WITHPREFIX q.fldname ;X in Control entry (col. 29)
                                      ;(PL/I compatible varchar field)
EXTRACT ITEMS ...,
      VARCHARMAX NOPREFIX q.fldname   ;Y in Control entry (col. 29)
                                      ;(Long character field)
```

Delimited Data Output Enhancements

Previously, when the CSV delimited data Report Method was specified, character-type fields were quoted whenever the field contained a comma (,) character. Sometimes, it was necessary to quote a field even when it did not contain an imbedded comma. For example, a character field containing all numeric characters but with leading zeros was assumed to be a numeric field rather than a character field when imported by Microsoft Excel unless the field was quoted. Excel discarded the leading zeros in the field such that the meaning and accurateness of the data might have been affected.

Additionally, when either the CSV or Tab delimited data Report Methods were specified, all trailing zeros for numeric fields with decimal places greater than one would be output, even if the zeros did not add any significance to the data value. If a field specification had many decimal places but most of the fields did not contain many significant digits following the decimal point, the size of the file became larger than necessary because of the cumulative number of non-significant zeros that may have occurred.

This enhancement addresses both of these issues and lets the application developer specify that a character-type field should always be quoted or that non-significant trailing zeros from a numeric-type field should be truncated. New Rn statement specifications are provided to specify new controls for delimited data output.

The new fixed-syntax specifications for the Rn statement Modifier entry (col. 37) are as follows:

Entry	Result
Q	Always enclose the character field within quotes (") regardless of whether it contains an imbedded comma (,) or not.
T	Truncate trailing zeros to the right of the decimal point of the numeric field up to either the decimal point (default), or up to the number of decimal places specified in the Percent/Ratio Field Decimal Places (col. 41) entry.

The new fixed-syntax specifications for the Rn statement Percent/Ratio Field Decimal Places entry (col. 41) are as follows:

Entry	Result
Blank	Percent or Ratio Field: <ul style="list-style-type: none"> ■ Ratio - 3 decimal places in the result. ■ Percent - 2 decimal places in the result. <p>Delimited data output field with T in Modifier entry (col. 37):</p> <ul style="list-style-type: none"> ■ Truncate trailing zeros up to but not including the decimal point.
0-9	Percent or Ratio Field: <ul style="list-style-type: none"> ■ 0-9 decimal places in the result. <p>Delimited data output field with T in Modifier entry (col. 37)</p> <ul style="list-style-type: none"> ■ Truncate trailing zeros up to the number of decimal places specified.

Prior to VISION:Builder Release 14.0, the Modifier entry (col. 37) was only used for Graph Reports.

The ASL examples below are the equivalent way of specifying the fixed-syntax specifications:

ITEM q.fldname CSVEDIT QUOTE ;	Always quote this field
ITEM q.fldname CSVEDIT TRUNCATE ;	Truncate all trailing 0's
ITEM q.fldname CSVEDIT TRUNCATE DECIMALS 2 ;	Truncate to 2 places

The following table illustrates the use of these specifications:

Specification	Data Type	Original Value	Output Value
None	Char	00010 West Hills, CA	00010 "West Hills, CA"
None	Numeric	123.05000 789.00000	123.05000 789.00000

Specification	Data Type	Original Value	Output Value
CSVEDIT QUOTE	Char	00010 West Hills, CA	"00010" "West Hills, CA"
CSVEDIT TRUNCATE	Numeric	123.05000 789.00000	123.05 789.
CSVEDIT TRUNCATE, DECIMALS 2	Numeric	123.05000 789.00000	123.05 789.00

HTML Primary Document Name Change

Previously, the primary object name for an HTML "report" was identified as MAIN. When there were many BDYnnnnn files or members associated with a "report", you may have had to scroll down into the file or member name list to locate the MAIN object. In VISION:Builder Release 14.0, the primary name has been changed to \$MAIN so that it always appears first in the list of files and members. This makes it easier for you to launch the browser for an HTML "report".

HFS Output for HTML Report

Previously, VISION:Builder required that HTML report output be placed into a Partitioned Data Set (PDS). A PDS is unique to MVS and could be accessed directly by WebSphere running on the OS/390 platform and served to a browser. However, when the HTML objects needed to be transferred to another platform (typically NT, LINUX, or UNIX) from which the browser would then access the "report", the .html suffix had to be added to each member name in the PDS to derive the target platform file name. Depending upon the file transfer tools used, this may have required manual intervention for each file transferred.

VISION:Builder Release 14.0 lets you output the HTML report into a Hierarchical File System (HFS) directory. HFS files have the same naming conventions as NT, LINUX, or UNIX files, and VISION:Builder Release 14.0 automatically appends the .html suffix to the object names whenever the destination file system is HFS rather than MVS. Then, if the files must be transferred to another platform, no renaming is required. The specification of HFS output is now a function of the JCL DD statement corresponding to the DDNAME specified for the HTML output. If the DD statement contains a PATHNAME= parameter, HFS output is performed. If the DD statement specifies a DSN= parameter, PDS output is performed.

Installing VISION:Builder

The VISION:Builder system is distributed on a cartridge tape, and contains sixteen files. The files contain:

1. JCL to copy the tape files 2-16 to disk.
2. CLIST members to run the install dialog.
3. ISPF panels for the install dialog.
4. ISPF messages for the install dialog.
5. ISPF JCL skeletons for the install dialog.
6. JCL members tailored by the install dialog.
7. VISION:Builder load modules.
8. Source statement and control members for customizing VISION:Builder.
9. COMLIB component load modules.
10. Workbench for ISPF component load modules.
11. Workbench for ISPF CLIST members.
12. Workbench for ISPF Panel members.
13. Workbench for ISPF Messages members.
14. Workbench for ISPF Skeleton members.
15. SAS/C[®] Runtime load modules.
16. Sample VISION:Builder applications.

Installation Tasks

Starting with release 14.0 of the VISION:Builder (and VISION:Two), the Installation and Maintenance will be managed by and under the control of the OS/390 SMP/E Facility as provided by IBM. This process differs significantly from previous releases (13.8 and prior) of VISION:Builder.

Additionally, the VISION:Builder Software System will use the CA License Management Program (LMP), which provides a standardized and automated approach to the tracking of licensed software.

You must perform the following basic tasks to install VISION:Builder:

1. Copy System Tape File 1 to a PDS
2. Copy System Tape Files 2 through 16 to Disk Data Sets
3. Complete the IP Dialog (create JCL Control Statements for the installation)
4. Allocate Data Sets
5. Define the CSI and the Global, Distribution, and Target Zones
6. RECEIVE the MCS and SYSMODS into the Global Zone
7. RECEIVE the PTF and APAR SYSMODS into the Global Zone
8. APPLY the VISION:Builder Elements (SYSMODS) to the Target Libraries
9. APPLY the VISION:Builder PTF SYSMODS to the Target Libraries
10. Run the Installation Verification Procedure using the Target Load Library.
11. ACCEPT the VISION:Builder Elements (SYSMODS) to the Distribution Libraries
12. ACCEPT the VISION:Builder PTF SYSMODS to the Distribution Libraries
13. APPLY Customizing APARs

The following tasks are optional. These tasks provide for customization and tailoring of the VISION:Builder Software based on specific site requirements and utilization criteria.

14. Customize the Parameter Modules
15. Install the DB2 Database Access Module MARKSQL
16. Install the PAL File Definitions and Requests
17. Relink Static Own Code Integration
18. Set Up for Use with the TSO Command Processor
19. Copy VISION:Builder Message Modules to LPA
20. Install VISION:Workbench for DOS
21. Set up VISION:Workbench for ISPF Requirements
22. Set up Quick Start Utility

For more information about installing VISION:Builder Release 14.0, see the [*VISION:Builder for OS/390 Installation Guide*](#).

VISION:Builder Quick Reference

This chapter lists some commonly used VISION:Builder applications which are described in the VISION:Builder books contained on this compact disc. This section describes where you would find information about those tasks and proposes several alternate solutions.

Use the online search function to locate the sections referenced in the following tables.

Specialized Report Formats

Task	Book	Chapter	Section
Report on preprinted forms.	VISION:Builder Reference Guide	Chapter 14	Formatted Reporting
Display data across the page instead of columnar.	VISION:Builder Reference Guide	Chapter 13	Reporting from Arrays
Generate labels.	VISION:Builder Reference Guide	Chapter 9	Report Page Layout
		Chapter 14	Formatted Reporting

Task	Book	Chapter	Section
Generate reports of non-standard sizes.	VISION:Builder Reference Guide	Chapter 14	Formatted Sectional Reporting Formatted Reporting Dynamic Report Line Modification
Display message on bottom of every page.	VISION:Builder Reference Guide	Chapter 14	Formatted Sectional Reporting


Reporting from Special Data Files

Task	Book	Chapter	Section
Summarize file data and change report information based on the use of summary data.	VISION:Builder Reference Guide	Chapter 14	Formatted Sectional Reporting
Report from multiple files.	VISION:Builder Reference Guide	Chapter 6	Coordinated Files
Report from multiple files when coordinating field is not contiguous.	VISION:Builder Reference Guide	Chapter 6 Chapter 15	Coordinated Files The Transaction Processing Step ⁴
Report failed transactions. ⁴	VISION:Builder Reference Guide	Chapter 15	Transaction Record Rejection and Type 4 Procedure Processing Flow

Task	Book	Chapter	Section
Report from header/trailer file.	VISION:Builder Reference Guide	Chapter 6	Coordinated Files
		Chapter 10	Transaction File and Master File Record Flows ⁴
		Chapter 14	Formatted Reporting
Report collating and routing to remote printers.	VISION:Builder Reference Guide	Chapter 17	Report Manager

Data Field and Record Processing

Task	Book	Chapter	Section
Sample Application.	VISION:Builder Reference Guide	Chapter 5	Sample Application Source Listing
Add fields to an existing file. ⁴	VISION:Builder Reference Guide	Chapter 5	Application Cycle Overview
		Chapter 10	Transaction File and Master File Record Flows
Clear out invalid data from numeric fields.	VISION:Builder Reference Guide	Chapter 4	Invalid Fields
		Chapter 10	Transaction Definitions ⁴
Perform calculations with time data.	VISION:Builder Reference Guide	Chapter 4	Time Data Conversions and Arithmetic Operations
Change the same field in every record on the file. ^B	VISION:Builder Reference Guide	Chapter 6	Master File Processing Options
		Chapter 10	Transaction File and Master File Record Flows

Task	Book	Chapter	Section
Procedurally update fields.	VISION:Builder Reference Guide	Chapter 6	Master File Processing Options
Remove selected records from a file. 	VISION:Builder ASL Reference Guide	Appendix C	Flags
	VISION:Builder Reference Guide	Chapter 7	Deleted Master File Records
		Chapter 10	Transaction File and Master File Record Flows
	VISION:Builder Specifications Guide	Chapter 3	Flags

File Manipulation

Task	Book	Chapter	Section
Read files with different key lengths.	VISION:Builder Reference Guide	Chapter 6	Coordinated Files
Process only a subset of the file.	VISION:Builder Reference Guide	Chapter 5	Application Cycle Overview
		Chapter 16	IMSTM Processing
	VISION:Builder Specifications Guide	Chapter 2	RC Statement
Compare keys in multiple files to determine keys on one file and not another.	VISION:Builder Reference Guide	Chapter 6	Coordinated Files
			Sequential Coordination
Convert a file from one database manager to another (VSAM to DB2).	VISION:Builder Reference Guide	Chapter 2	Discipline of VISION:Builder
		Chapter 3	Define a Relational File
		Chapter 7	Subfiles

Task	Book	Chapter	Section
Audit trail when updating. ⁴	VISION:Builder Reference Guide	Chapter 10	Transaction File and Sequential Master File Record Flows
		Chapter 15	Transaction Record Rejection and Type 4 Procedure Processing Flow
Read IMS and DB2 files in same run.	VISION:Builder Reference Guide	Chapter 3	Concept of Structured Files
			Define an IMS File
		Chapter 6	Define a Relational File
Control reading of a file.	VISION:Builder Reference Guide	Chapter 6	Coordinated Files
		Chapter 13	Controlling Array Looping
Generate a test file from a production file.	VISION:Builder Reference Guide	Chapter 7	Subfiles
Stop processing on demand.	VISION:Builder Reference Guide	Chapter 5	Flag Fields
		Chapter 8	Branching to Control Looping
		Chapter 10	Transaction File and Master File Record Flows ⁴
Process table databases.	VISION:Builder Reference Guide	Chapter 3	Define a Relational File
		Chapter 15	Relational Updating Considerations
Restrict users view of file data.	VISION:Builder Reference Guide	Chapter 3	Using a Logical Record

Task	Book	Chapter	Section
Check program for efficiency.	VISION:Builder Reference Guide	Chapter 20	Using PAL to Help Maintain a Program
	VISION:Builder Environment Guide		
Feed parameter to VISION:Builder program (EXEC linkage or user-read).	VISION:Builder Reference Guide	Chapter 6	User Coordination
		Chapter 18	entire chapter
	VISION:Builder Environment Guide	Chapter 6	entire chapter

Definition Processing

Task	Book	Chapter	Section
Use a different definition - just for this run.	VISION:Builder Reference Guide	Chapter 3	Catalog the File Definition in the Common Library
		Chapter 19	Processing with Multiple Common Libraries
How to use multiple COMLIBs.	VISION:Builder Reference Guide	Chapter 19	Instream File Definitions
	VISION:Builder Environment Guide	Chapter 13	entire chapter

COBOL and VISION:Builder

Task	Book	Chapter	Section
Access COBOL file with OCCURS DEPENDING ON followed by more fields.	VISION:Builder Reference Guide	Chapter 4	Variable Length Fields
		Chapter 18	entire chapter
Call a COBOL routine from VISION:Builder.	VISION:Builder Reference Guide	Chapter 18	entire chapter

Table Processing

Task	Book	Chapter	Section
Convert codes to descriptions automatically.	VISION:Builder Reference Guide	Chapter 12	Defining Tables
			Automatic Table Lookup

Performance Tuning

Task	Book	Chapter	Section
Optimize your program.	VISION:Builder Reference Guide	Chapter 20	entire chapter

Note: This chapter is applicable to OS/390 only.

COBOL quick start is a batch utility that generates a skeletal COMLIB file definition from an existing COBOL file definition. Once a skeletal COMLIB file definition has been created, any additional file information that is required can be specified using the VISION:Workbench for DOS, VISION:Workbench for ISPF, or a text editor.

The COBOL quick start utility can also run from VISION:Workbench for ISPF. The IMPORT Option points you to a menu for selecting the quick start utility you want to run. The subsequent panels prompt you for the information needed to run the utility. Once the information is gathered, the utility is run immediately and the output displayed for you to browse.

Note: File definitions are stored to and retrieved from a common library (COMLIB) when needed at processing time. Unless specifically stated otherwise, the term COMLIB refers to this common file definition.

The COBOL quick start Utility can retrieve COBOL copybooks from OS/390® data sets, CA-Panvalet® libraries, and CA-Librarian® libraries. Most COMLIB file definition types are supported by COBOL quick start.

Flow Diagram

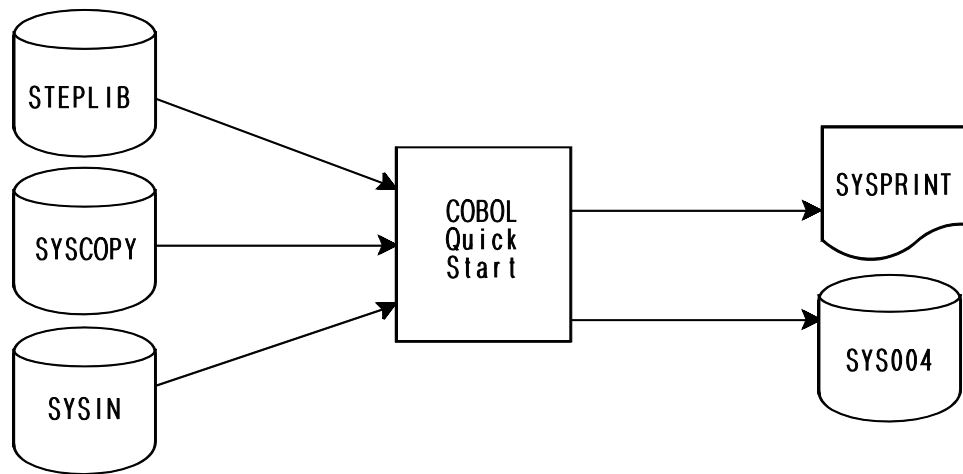


Figure 5-1 COBOL Quick Start Flow Diagram

Note: If a CA-Panvalet or CA-Librarian copybook library is used, a PANDD1 (CA-Panvalet) or MASTER (CA-Librarian) DD statement is required. See [Using CA-Panvalet and CA-Librarian COBOL Copybooks on page 5-4](#) for more information.

As shown in [Figure 5-1](#), COBOL quick start uses the following input data sets:

- STEPLIB Specifies the COMLIB load library.
- SYSCOPY Specifies an OS/390 COBOL copybook library.
- SYSIN Provides the appropriate COBOL quick start control statements. Instream COBOL field definitions may also be specified here.

COBOL quick start produces two output files:

- SYSPRINT Contains a report on the file definition generation process which includes a listing of the COBOL statements that were processed.
- SYS004 Contains the generated COMLIB file definition source statements.

Utility Execution

COBOL quick start is a batch utility. The Job Control Language (JCL) statements required to execute this utility are shown in [Figure 5-2](#). The JCL contains an instream procedure, followed by JCL statements to execute the procedure. Notice that this JCL uses sample SYSIN data. You can use this sample COBOL data to see exactly how COBOL quick start works.

At a minimum, you must make the following changes before submitting this JCL:

- Supply a JOB Card.
- Supply values for the procedure variables detailed in the following table.

Variable Name	Description
CLLOAD	Specify the name of your COMLIB load library.
COPYLIB	Specify the name of the COBOL copylib that is needed.
DEFLIB	Specify the source definition library where the new file definition should be written.
MEMBER	Specify a member name for the new file definition. This name must be the same name that is specified in the NAME parameter on the FILEGEN statement. If an existing member name is specified, it is overwritten. See the specifications for the FILEGEN control statement in FILEGEN Control Statement on page 5-7 for more details.

- Provide a DD statement override for COBOLQS.SYSIN. This data set contains the required COBOL quick start input control statements used to control the generation of the COMLIB file definition. See [Control Statements on page 5-7](#) for detailed information on these control statements.
- To conform to your shop standards, additional modifications may be required.

```

/* MEMBER CLCOBQS
/******
/* EXECUTE THE COBOL QUICK START UTILITY.
/* THE SYSCOPY DD STATEMENT IS USED FOR MVS COPYBOOK LIBRARIES.
/* THE PANDD1 DD STATEMENT IS USED FOR PANVALET COPYBOOK LIBRARIES.
/* THE MASTER DD STATEMENT IS USED FOR LIBRARIAN COPYBOOK LIBRARIES.
/******
//COBOLQS PROC CLLOAD=,
//              COPYLIB=,
//              DEFLIB=,
//              MEMBER=
//COBOLQS EXEC PGM=COBOLQS,REGION=1024K
//STEPLIB DD DSN=&CLLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSCOPY DD DSN=&COPYLIB,DISP=SHR
//PANDD1 DD DSN=&COPYLIB,DISP=SHR
//MASTER DD DSN=&COPYLIB,DISP=SHR

```

Figure 5-2 Sample Execution JCL for the COBOL Quick Start Utility (Page 1 of 2)

```
//SYS004 DD DSN=&DEFLIB(&MEMBER),DISP=OLD
//SYSIN DD DUMMY
// PEND
//*****
//* BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING
//* INFORMATION:
//* CLLOAD - NAME OF YOUR COMLIB LOAD LIBRARY
//* COPYLIB - NAME OF YOUR COBOL COPY LIBRARY. THIS IS AN
//* MVS, PANVALET, OR LIBRARIAN COPYBOOK LIBRARY.
//* DEFLIB - NAME OF YOUR COMLIB SOURCE DEFINITION LIBRARY
//* THE GENERATED FILE DEFINITION IS WRITTEN TO
//* THIS LIBRARY.
//* MEMBER - MEMBER NAME FOR THE DEFINITION YOU ARE GENERATING.
//*
//* YOU MUST ALSO PROVIDE THE APPROPRIATE SYSIN DATA IN THE
//* COBOLQS.SYSIN DD STATEMENT OVERRIDE.
//*****
//QS EXEC COBOLQS,
// CLLOAD='BUILDER.CL045.LOADLIB',
// COPYLIB='COBOL.COPYBOOK.LIBRARY',
// DEFLIB='COMLIB.DEFLIB',
// MEMBER='SAMPLEFD'
//COBOLQS.SYSIN DD *
FILEGEN NAME=SAMPLEFD,TYPE=FIXED,RECSIZE=80
SEGMENT NAME=OFFICE,NUMBER=10,LEVEL=1
$COBOL
01 OFFICE-DATA.
02 OFFICE-CODE PIC S9(3).
02 OFFICE-ADDRESS.
03 OFFICE-STREET PIC X(20).
03 OFFICE-CITY PIC X(15).
03 OFFICE-STATE PIC X(2).
03 OFFICE-ZIP.
04 OFFICE-ZIP-FIRST-FIVE PIC X(5).
04 OFFICE-ZIP-LAST-FOUR PIC X(4).
02 OFFICE-PHONE PIC 9(7).
02 OFFICE-AREA-CODE PIC X(3).
02 SPEED-DIAL PIC X(3).
02 FILLER PIC X(18).
$ECOBOL
/*
```

Figure 5-2 Sample Execution JCL for the COBOL Quick Start Utility (Page 2 of 2)

Using CA-Panvalet and CA-Librarian COBOL Copybooks

COBOL quick start provides direct access to COBOL copybooks that are stored in CA-Panvalet or CA-Librarian source libraries. Before using this facility, you must link edit the appropriate CA-Panvalet or CA-Librarian interface modules with the COBOL quick start interface modules. The required CA-Panvalet and CA-Librarian interface modules are included in your CA-Panvalet or CA-Librarian software package.

CA-Panvalet Interface

Note: The generated COMLIB field length for a GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC string is 2*n, where n is the DB2 length of the string (number of DBCS characters).

[Figure 5-3](#) contains the JCL required for link editing the CA-Panvalet interface modules with the COBOL quick start load module.

You may have to replace the INCLUDE LIBSYS(PAM) statement with several INCLUDE statements. See your CA-Panvalet book for the exact INCLUDE statements that are required.

```

/* MEMBER CLCOBPL
/******
/* LINK EDIT PANVALET INTERFACE MODULES WITH COBOL QUICK START.
/******
//CLPANLK PROC CLLOAD=,
//      PANLOAD=
//LINK   EXEC PGM=IEWL,REGION=512K,PARM='LIST,MAP,LET,XREF,NCAL'
//SYSLIB DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//LIBSYS DD DSN=&PANLOAD,DISP=SHR
//LLIB   DD DSN=&CLLOAD,DISP=SHR
//SYSLMOD DD DSN=&CLLOAD,DISP=SHR
//      PEND
/******
/* BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING
/* INFORMATION:
/*      CLLOAD - NAME OF YOUR COMLIB LOAD LIBRARY.
/*      PANLOAD - NAME OF YOUR PANVALET SYSTEM LOAD LIBRARY.
/******
//PANLINK EXEC CLPANLK,
//      CLLOAD='BUILDER.CL045.LOADLIB',
//      PANLOAD='PANVALET.SYSTEM.LOADLIB'
//LINK.SYSLIN DD *
//      INCLUDE LIBSYS(PAM)
//      INCLUDE LLIB(COMLIBP)
//      ENTRY COMLIBP
//      NAME COMLIBP(R)
/*

```

Figure 5-3 JCL to Link CA-Panvalet Interface Modules

After the CA-Panvalet interface modules have been link edited, the COPYPCOBOL parameter on the SEGMENT statement can be used. [SEGMENT Control Statement on page 5-9](#) contains the specifications for the SEGMENT statement.

CA-Librarian Interface

[Figure 5-4](#) contains a listing of the JCL required for link editing the CA-Librarian interface modules with the COBOL quick start interface modules.

You may have to replace the INCLUDE LIBSYS(FAIR) statement with several INCLUDE statements. See your CA-Librarian book for the exact INCLUDE statements that are required. Unresolved external references will occur when link editing the CA-Librarian interface; these are normal and can be safely ignored.

```

/* MEMBER CLCOBLL                                00010000
/*****
00020000
/* LINK EDIT LIBRARIAN INTERFACE MODULES WITH COBOL QUICK START.  * 00030000
/*****
00040000
//CLLIBLK PROC CLLOAD=,                                00050000
//                                LIBLOAD=                00060000
//LINK EXEC PGM=IEWL,REGION=2M,PARM='LET,LIST,MAP,NCAL'  00070000
//SYSLIB DD DUMMY                                00080000
//SYSPRINT DD SYSOUT=*                            00090000
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))            00100000
//LIBSYS DD DSN=&LIBLOAD,DISP=SHR                    00110000
//LLIB DD DSN=&CLLOAD,DISP=SHR                      00120000
//SYSLMOD DD DSN=&CLLOAD,DISP=SHR                    00130000
// PEND                                              00140000
/*****
00150000
/* BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING * 00160000
/* INFORMATION:                                           * 00170000
/*                                                         * 00180000
/* CLLOAD - NAME OF YOUR COMLIB LOAD LIBRARY.            * 00190000
/* LIBLOAD - NAME OF YOUR LIBRARIAN SYSTEM LOAD LIBRARY. * 00200000
/*                                                         * 00210000
/*****
00220000
//LIBLINK EXEC CLLIBLK,                                00230000
//                                CLLOAD='BUILDER.CL045.LOADLIB', 00240000
//                                LIBLOAD='LIBRARN.SYSTEM.LOADLIB' 00250000
//LINK.SYSLIN DD *                                00250100
INCLUDE LIBSYS(FAIRCLS)                            00250200
INCLUDE LIBSYS(FAIRERR)                            00250300
INCLUDE LIBSYS(FAIRLOC)                            00250400
INCLUDE LIBSYS(FAIRMOD)                            00250500
INCLUDE LIBSYS(FAIRNTE)                            00250600
INCLUDE LIBSYS(FAIROPN)                            00250700
INCLUDE LIBSYS(FAIRPNT)                            00250800
INCLUDE LIBSYS(FAIRREC)                            00250900
INCLUDE LIBSYS(FAIRSCAN)                           00251000
INCLUDE LIBSYS(FAIRSEC)                            00251100
INCLUDE LLIB(COMLIB)                                00251200
ENTRY COMLIB                                         00251300
NAME COMLIB(R)                                      00252000
/*                                                    00260000

```

Figure 5-4 JCL to Link CA-Librarian Interface Modules

After the CA-Librarian interface modules have been link edited, the COPYLCOBOL parameter on the SEGMENT statement can be used. [SEGMENT Control Statement on page 5-9](#) contains the specifications for the SEGMENT statement.

Control Statements

The COBOL quick start Utility uses the following control statements as input:

- **FILEGEN** The FILEGEN statement provides a name for the file definition that is being generated and identifies the type of file (for example, VSAM KSDS or IMS) being generated.
- **SEGMENT** The SEGMENT statement is used to define the segments within a file.
- **\$COBOL** This statement signals that an in-stream COBOL definition follows.
- **\$ECOBOL** This statement signals the end of an in-stream COBOL definition.

The function and syntax of these statements are described in the following sections.

Coding Rules

When writing COBOL quick start control statements, the following rules must be observed:

- Each non-continued control statement must contain a control statement command that identifies the control statement type.
- Each control statement may contain keyword parameters. Keyword parameters may be specified in any order. Keyword parameters must be separated by commas. Embedded blanks are not allowed between parameters.
- Each parameter must be coded unless stated otherwise.
- A comma following the last parameter on a statement causes a continuation to the next statement.
- Comments may be placed after the last parameter with an intervening blank.
- Columns 1 through 71 of the control statement are scanned. Columns 72 through 80 are ignored.

FILEGEN Control Statement

Statement Syntax

```
FILEGEN  NAME= ,
          TYPE=,
          RECSIZE=,
          RECBLK=,
          BUFFSIZE=,
          FLDPREFX=
```

Figure 5-5 FILEGEN Control Statement Format

- **FILEGEN**
(required) FILEGEN is a control statement command. It identifies the control statement type.
- **NAME**
(required) The NAME parameter specifies the name of the file definition being generated. A file name can be from 1-8 characters long. The first character must be alphabetic. The remaining characters can be a combination of alphanumeric characters. When the generated file definition is written to a partitioned data set, the file name specified here must be identical to the member name specified in the JCL.
- **TYPE**
(required) The TYPE parameter specifies the type of COMLIB file definition that you want to create. Valid file types are listed in the following table.

File Type	Description
DB2	DB2 Relational Database
KSDS	VSAM Key Sequenced Data Set
ESDS	VSAM Entry Sequenced Data Set
AIX	VSAM Alternate Index Data Set
DLI	IMS Database
DLIHDAM	IMS HDAM Database
ISAMFIX	ISAM Fixed Length Record Format Data Set
ISAMVAR	ISAM Variable Length Record Format Data Set
FIXED	Fixed Length Record Format Data Set
VARIABLE	Variable Length Record Format Data Set
UNDEFINED	Undefined Record Format Data Set
GDBI	General Data Base Interface Mapped File

- **RECSIZE**
(optional) The RECSIZE parameter specifies the number of data bytes in the data portion of a record or segment. Enter a number from 1-9999. This parameter only applies to FIXED, ISAMFIX, VARIABLE, or ISAMVAR file types.
- **RECBLK**
(optional) The RECBLK parameter specifies the number of records in each block. Enter a number from 1-999. This parameter only applies to FIXED and ISAMFIX file types.

- **BUFSIZE**
(optional) The BUFSIZE parameter specifies the size of the buffer that is needed to process the file. Enter a value from 1-32760 or 1K-9999K.

For DB2, DLI, DLIHDAM, and GDBI file types, enter the maximum amount of main storage required to hold a logical record. For KSDS and ESDS file types, enter the maximum record size according to the VSAM cluster definition. If the file type is AIX, enter the alternate index control interval size. For ISAMFIX, ISAMVAR, FIXED, VARIABLE, and UNDEFINED, enter the block size.

- **FLDPREFIX**
(optional) The FLDPREFIX parameter specifies the 1-3 character prefix to be used for generating primary field names in the COMLIB file definition. Primary field names are required in a COMLIB file definition and must be assigned a unique 1-8 character name. Since COBOL field names can be longer than 8 characters and may not be unique within the first 8 characters, COBOL quick start automatically generates a unique 8-character primary field using the FLDPREFIX value followed by a generated field number.

If the FLDPREFIX parameter is omitted, the default prefix is F and the generated primary field names have the format Fnnnnnnnn where nnnnnnnn is a number from 0000001-9999999. See [COMLIB Field Name Generation on page 5-12](#) for more information.

SEGMENT Control Statement

Statement Syntax

```
SEGMENT  NAME= ,
          NUMBER=,
          LEVEL=,
          COPYCOBOL=,
          COPYPCOBOL=,
          COPYLCOBOL=
```

Figure 5-6 SEGMENT Control Statement Format

- **SEGMENT**
(required) SEGMENT is a control statement command. It identifies the control statement type.
- **NAME**
(required) The NAME parameter assigns a name to a segment. Segment names can be from 1-8 characters. The first character must be alphabetic. The remaining characters can be a combination of alphanumeric characters.

- | | |
|----------------------------|---|
| ■ NUMBER
(required) | The NUMBER parameter assigns a number that uniquely identifies the segment. Enter a number from 1-255. Subordinate segments must have a number larger than the parent segment and smaller than any children segments. |
| ■ LEVEL
(required) | The LEVEL parameter specifies the subordination of segments. The root segment must have a level number of 1. All subordinate segments must be assigned a number from 2 to 9. |
| ■ COPYCOBOL
(optional) | The COPYCOBOL parameter specifies the name of the COBOL copybook that contains the field definitions for this segment. This copybook must be located in an OS/390 data set assigned to the SYSCOPY DD statement in the JCL. |
| ■ COPYPCOBOL
(optional) | The COPYPCOBOL parameter specifies the name of the COBOL copybook that contains the field definitions for this segment. This copybook must be located in a CA-Panvalet library assigned to the PANDD1 DD statement in the JCL. |
| ■ COPYLCOBOL
(optional) | The COPYLCOBOL parameter specifies the name of the COBOL copybook that contains the field definitions for this segment. This copybook must be located in a CA-Librarian library assigned to the MASTER DD statement in the JCL. |

Processing Notes

COPYCOBOL, COPYPCOBOL, and COPYLCOBOL parameters are mutually exclusive. Only one of these parameters should be specified on a SEGMENT statement. [Figure 5-7](#) shows an example of the input control statements used with the COPY parameter.

```
FILEGEN NAME=XXXX,TYPE=X  
SEGMENT NAME=XXXX,NUMBER=XX,LEVEL=X,  
COPYCOBOL=copybookname
```

Figure 5-7 Control Statements Used with the COPY Parameter

When using any of the COPY parameters (COPYCOBOL, COPYPCOBOL, COPYLCOBOL), a NOPRINT option can be specified to suppress the listing of the copybook on the SYSPRINT file. This option is specified as
`COPYCOBOL=(copybookname,NOPRINT)`

If a COPY parameter is not specified on the SEGMENT statement, then an in-stream COBOL definition must be provided. In this case, the SEGMENT statement must be followed by a \$COBOL statement, the in-stream COBOL definition, and the \$ECOBOL statement which marks the end of the definition. See [\\$COBOL and \\$ECOBOL Control Statements on page 5-11](#) for more information.

\$COBOL and \$ECOBOL Control Statements

Example of Syntax and Use

```
FILEGEN NAME=XXXX,TYPE=X  
SEGMENT NAME=XXXX,NUMBER=XX,LEVEL=X  
$COBOL  
instream COBOL source statements  
$ECOBOL
```

Figure 5-8 \$COBOL and \$ECOBOL Control Statements

The \$COBOL and \$ECOBOL control statements are used in place of the SEGMENT COPY parameter to process instream COBOL source statements. The \$COBOL statement must follow a SEGMENT statement that does not contain a COPY parameter. The actual COBOL source statements to be processed must follow the \$COBOL statement. The end of the in-stream COBOL source statements must be marked by the \$ECOBOL statement. The in-stream COBOL source statements are used to generate field definitions for the segment.

\$COBOL and \$ECOBOL can be placed anywhere within columns 1-71. They must be the only command on the statement. There are no parameters for either statement.

Conversion Rules

Generated COMLIB File Definition

This utility generates a skeletal COMLIB file definition which must be edited and validated by VISION:Workbench for DOS, VISION:Workbench for ISPF, or a text editor prior to its use. The following items should be considered when editing the generated file definition:

- | | |
|--------------------------|---|
| ■ Segment key assignment | Each segment must define at least one field as the key; additional keys can be assigned if wanted. |
| ■ Segment information | Additional segment information such as segment order and number of fixed occurrences should be provided as needed. |
| ■ Field information | Primary and alternate field name assignments can be modified if wanted (see COMLIB Field Name Generation on page 5-12). Additional field information such as rounding, editing, and automatic table lookup results should be provided as needed. |

COMLIB Field Name Generation

All fields within a COMLIB file definition must be assigned a unique 1-8 character name. This name is referred to as the primary field name.

When building a skeletal COMLIB file definition from COBOL field definitions, a unique 1-8 character primary field name must be assigned to each field. Since COBOL field names can be longer than 8 characters and may not be unique within the first 8 characters, COBOL quick start automatically generates a unique primary field name using the FLDPREFX parameter value on the FILEGEN statement followed by a generated field number. If the FLDPREFX parameter is omitted, the default prefix is F and the generated primary field names have the format Fnnnnnnnn where nnnnnnnn is a number from 0000001-9999999.

COBOL quick start uses the COBOL field name to generate the Column Heading specifications.

Once the skeletal file definition has been created, it can be imported into the VISION:Workbench for DOS, VISION:Workbench for ISPF, and the generated primary field names can be replaced with more descriptive primary field names.

Unsupported COBOL Specifications

COBOL quick start does not support the following COBOL field specifications:

- 66 levels. All references to the field are removed. A warning message is issued.
- 77 levels. Any field defined at level 77 will have a starting location of 1. The resulting definition should be checked. A warning message is issued.
- 88 levels. All references to the field are removed. A warning message is issued.
- Numeric fields with more than 9 digits to the right of the decimal place are supported for packed fields; however, for any other field type, a warning message is issued indicating that the number of decimal places has been truncated to 9.
- Binary fields larger than S9(9) are generated as character fields. A warning message is issued.
- COMP-2 data types are generated as 8-byte character fields. A warning message is issued.
- If the length of a numeric field exceeds 15 digits, the field is generated as character. A warning message is issued.
- The P edit parameter on the PICTURE clause always generates a zero SCALE value. The resulting definition should be checked. A warning message is issued.
- If the size of a field exceeds 255, a warning message is issued indicating that the generated field length has been truncated to 255. The generated definition should be modified to include an additional field that defines the remaining bytes or at least the last byte of the field.
- OCCURS clause. A warning message is issued to indicate that only 1 occurrence of the item was generated. Additional occurrences must be added to the definition.

If the item on the OCCURS clause is variably occurring (DEPENDING ON clause present), then the additional occurrences can be defined by defining the generated occurrence as a lower level variably occurring segment, if possible. This requires the field containing the number of occurrences to be defined as a count field in the generated file definition.

When a DEPENDING ON clause is encountered, an additional message is issued to inform you that you must adjust the field start locations of subsequent fields. This is because the generated start location of the next field will be incorrect if you define the generated occurrence as a lower level variably occurring segment. Also, the start location of the next field is calculated using the maximum number of occurrences when in reality the number of occurrences varies.

If the item on the OCCURS clause is fixed occurring (no DEPENDING ON clause), then the additional occurrences can be defined either by:

- Defining the generated occurrence as a lower level fixed occurring segment.

If this method is used, a reference to the field will initiate a loop where each occurrence of the field is automatically processed.

- Defining a separate field for each occurrence.

If this method is selected, then a separate field name exists for each occurrence.

Note: If the field name is FILLER or blank, all references to the field are removed from the definition.

- Defining one field whose length accommodates all occurrences (for example, define a 30-byte field if you have a 3-byte field that occurs 10 times).

If this method is chosen, then each occurrence can be accessed by using dynamic partial fielding, which is similar to indexing.

Note: This chapter is applicable to OS/390 only.

DB2 quick start is a batch utility that generates COMLIB file definitions from existing DB2 table definitions. During DB2 quick start processing, each DB2 table that is processed becomes a separate segment in the COMLIB file definition being created. DB2 column information is retrieved from the DB2 SYSCOLUMNS table, converted into COMLIB field specifications, and added to the segment being created. A COMLIB field statement (L0) is created for each column in the specified DB2 table.

Multiple file definitions can be generated in a single DB2 quick start execution. Each generated file definition is stored as a separate member in the indicated source definition library. The specified file name is used as the member name when the generated definition is saved in the source definition library. If an existing member name is specified, the existing member is replaced when the new definition is saved. If a new member name is specified, then a new member is created.

Once a COMLIB file definition has been created, any additional file information that is required can be added with the VISION:Workbench for DOS or ISPF, or a text editor.

The DB2 quick start utility can also run from VISION:Workbench for ISPF. The IMPORT Option points you to a menu for selecting the quick start utility you want to run. The subsequent panels prompt you for the information needed to run the utility. Once the information is gathered, the utility is run immediately and the output displayed for you to browse.

Note: File definitions are stored to and retrieved from a common library (COMLIB) when needed at processing time. Unless specifically stated otherwise, the term COMLIB refers to this common file definition.

Flow Diagram

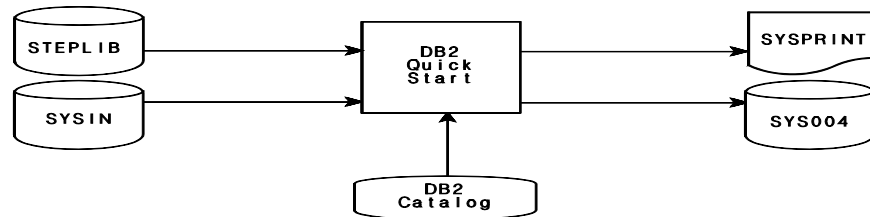


Figure 6-1 DB2 Quick Start Flow Diagram

As shown in [Figure 6-1](#), DB2 quick start uses the following two input data sets:

- STEPLIB Specifies the COMLIB load library and the DB2 load library.
- SYSIN Specifies the appropriate DB2 quick start control statements.

DB2 quick start produces two output files:

- SYSPRINT Contains a summary report on the file definition generation process. The PRINT parameter on the SEGMENT control statement can be used to control the contents of this report. See [SEGMENT Control Statement on page 6-8](#) for more information.
- SYS004 Generated file definitions are written to the partitioned data set pointed to by the DD name SYS004. This must be a partitioned data set.

Utility Execution

DB2 quick start is a batch utility. Before this utility can be executed, the DB2 quick start Data Base Request Module (DBRM), provided on the installation tape, must be input to a DB2 bind. This creates the DB2 plan required by the DB2 quick start utility.

[Figure 6-2 on page 6-3](#) contains the same JCL required to execute DB2 quick start. This JCL contains an in-stream procedure followed by JCL statements to execute the procedure and sample DB2 quick start control statements. The sample control statements can be used to create a COMLIB file definition from the sample tables provided with DB2 (DSN8230.DEPT, DSN8230.EMP, and DSN8230.PROJ). If you have created a DB2 plan by binding the DB2 quick start DBRM during the installation process, you can use these sample control statements to see how DB2 quick start works.

At a minimum, you must make the following changes before submitting this JCL:

- Supply a JOB card
- Supply values for the following procedure variables detailed in the following table.

Variable Name	Description
CLLOAD	Specify the name of your COMLIB load library.
DB2LOAD	Specify the name of your DB2 load library.
DEFLIB	Specify the source definition library to which the new file definition should be written. DEFLIB must specify a partitioned data set.

- Provide a DD statement override for DB2QS.SYSIN. This data set contains the required DB2 quick start input control statements which are used to control the generation of COMLIB file definitions. See [Control Statements on page 6-4](#) for detailed information on these control statements.
- To conform to your shop standards, additional modifications may be required.

```

/* MEMBER CLDB2QS
/******
/* EXECUTE THE DB2 QUICK START UTILITY.
/******
//DB2QS  PROC CLLOAD=,
//          DB2LOAD=,
//          DEFLIB=
//DB2QS  EXEC PGM=DB2QS,REGION=1024K
//STEPLIB DD DSN=&CLLOAD,DISP=SHR
//          DD DSN=&DB2LOAD,DISP=SHR
//SYSTEM  DD DUMMY
//SYSPRINT DD SYSOUT=*,
//          DCB=(DSORG=PS,RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYS004  DD DSN=&DEFLIB,DISP=OLD
//SYSIN   DD DUMMY
//          PEND
/******
/* BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING
/* INFORMATION:
/*
/*      CLLOAD - NAME OF YOUR COMLIB LOAD LIBRARY.
/*      DB2LIB - NAME OF YOUR DB2 DSN.DSNLOAD LIBRARY.
/*      DEFLIB - NAME OF YOUR COMLIB SOURCE DEFINITION LIBRARY.
/*              THE GENERATED FILE DEFINITION IS WRITTEN TO
/*              THIS LIBRARY.
/*
/* YOU MUST ALSO PROVIDE THE APPROPRIATE SYSIN DATA IN THE
/* DB2QS.SYSIN DD STATEMENT OVERRIDE.
/******

```

Figure 6-2 Sample Execution JCL for the DB2 Quick Start Utility (Page 1 of 2)

```
//QS      EXEC DB2QS,
//          CLLOAD='BUILDER.CL045.LOADLIB',
//          DB2LOAD='DB2.SYSTEM.DSNLOAD',
//          DEFLIB='COMLIB.DEFLIB'
// *
//DB2QS.SYSIN DD *
DB2CNTL DB2PLAN=DB2QS,DB2SYS=DB2T
FILEGEN NAME=DB2FD,BUFFSIZE=1024K
SEGMENT NAME=DEPT,NUMBER=10,LEVEL=1,TABLE=DEPT,CREATOR=DSN8230,
        PRINT=ALL
NEWPAGE
SEGMENT NAME=EMPLOYEE,NUMBER=20,LEVEL=2,TABLE=EMP,CREATOR=DSN8230,
        PRINT=ALL
NEWPAGE
SEGMENT NAME=PROJECT,NUMBER=30,LEVEL=2,TABLE=PROJ,CREATOR=DSN8230,
        PRINT=ALL
/*
```

Figure 6-2 Sample Execution JCL for the DB2 Quick Start Utility (Page 2 of 2)

Control Statements

The DB2 quick start utility uses the following control statements as input:

- **DB2CNTL** Information from the DB2CNTL statement establishes the proper DB2 environment.
- **FILEGEN** The FILEGEN statement triggers the start of a new file definition. The NAME parameter specified on the FILEGEN statement is used as both the file name and the member name when the generated file definition is saved.
- **SEGMENT** The SEGMENT statement defines file segments. Each segment corresponds to a DB2 table. Field information for the specified DB2 table is retrieved from the DB2 SYSCOLUMNS table, converted to COMLIB field specifications, and used to create field definitions for the current segment.
- **NEWPAGE** The NEWPAGE statement triggers a page eject on the DB2 quick start summary report.

The function and syntax of these statements are described in the following sections.

Coding Rules

When writing DB2 quick start control statements, the following rules must be observed:

- Each non-continued control statement must contain a control statement command that identifies the control statement type.
- Each control statement may contain keyword parameters. Keyword parameters may be specified in any order. Keyword parameters must be separated by commas. Embedded blanks are not allowed between parameters.
- Each parameter must be coded unless stated otherwise.
- A comma following the last parameter on a statement causes a continuation to the next statement.
- Comments may be placed after the last parameter with an intervening blank.
- Blank lines within the SYSIN file are printed on the summary report output to SYSPRINT. Blank lines, along with the NEWPAGE control statement can be used to format the DB2 quick start summary report.
- Columns 1 through 71 of the control statement are scanned. Columns 72 through 80 are ignored.

DB2CNTL Control Statement

Statement Syntax

```
DB2CNTL  DB2PLAN=,  
          DB2SYS=
```

Figure 6-3 DB2CNTL Control Statement Format

- **DB2CNTL (required)** DB2CNTL is a control statement command. It identifies the control statement type. The DB2 control statement is used to establish the proper DB2 environment. DB2 quick start uses the TSO Call Attach Facility to connect to DB2. Static SQL is used to process information in the DB2 SYSCOLUMNS table.
- **DB2PLAN (required)** The DB2PLAN parameter specifies the name of the DB2 plan that should be used when accessing DB2 tables. The DB2 plan is created during the installation process by binding the provided DB2 quick start DBRM.
- **DB2SYS (required)** The DB2SYS parameter specifies the name of the DB2 system that is to be used.

FILEGEN Control Statement

Statement Syntax

```
FILEGEN  NAME=,  
          BUFFSIZE=,  
          FLDPREFIX=,  
          FLDNAME=,  
          HEADING=,  
          LOGREL=,  
          LONGNAME=,  
          DESCRIPT=,  
          DATEFLD=
```

Figure 6-4 FILEGEN Control Statement Format

- **FILEGEN**
(required) FILEGEN is a control statement command. It identifies the control statement type. The FILEGEN control statement triggers the start of a new file definition.
- **NAME**
(required) The NAME parameter specifies the name of the file definition that is being generated. A file name can be from 1 to 8 characters. The first character must be alphabetic. The remaining characters can be a combination of alphanumeric characters.

When the generated file definition is saved, the file name is also used as the member name. If an existing member name is specified, the existing member is overwritten. If a new member name is specified, a new member is created.
- **BUFFSIZE**
(optional) The BUFFSIZE parameter defines the size of the buffer needed to process a logical record. Enter a number from 1 to 32760. Multiples of 1024 can be entered as nnnnK where nnnn is a number between 1 to 9999.
- **FLDPREFIX**
(optional) The FLDPREFIX parameter specifies the 1 to 3 character prefix for generating primary field names in the COMLIB file definition. Primary field names are required in a COMLIB file definition and must be assigned a unique 1 to 8-character name. Since DB2 field names can be longer than 8-characters and may not be unique within the first 8 characters, DB2 quick start automatically generates a unique 8-character primary field using the FLDPREFIX value followed by a generated field number.

If the FLDPREFIX parameter is omitted, the default prefix is F and the generated primary field names have the format Fnnnnnnnn where nnnnnnnn is a number from 0000001 to 9999999. See [COMLIB Field Name Generation on page 6-10](#) for more information.

- **FLDNAME**
(optional) The FLDNAME parameter specifies how the field name (short name) is derived. Enter GEN to specify that the name is to be generated using the FLDPREFIX parameter convention. Enter TRUNC to specify that the name is to be a truncation (first 8 characters) of the DB2 table column name.

If the FLDNAME parameter is omitted, an entry of GEN is assumed. Note that an entry of TRUNC can result in duplicate names that you need to resolve by modifying the generated code.
- **HEADING**
(optional) The HEADING parameter specifies whether the Column Heading specifications (Ln statements) in the generated definition should contain the DB2 Label information for the SYSCOLUMNS table or the DB2 column name. Enter LABEL if the label information is to be used or COLUMN if the column name is to be used. If the HEADING parameter is omitted, the LABEL specification is assumed.
- **LOGREL**
(optional) The LOGREL parameter specifies whether the DB2 Quick Start utility should generate LR statements for the definition. Enter YES to cause LR statements to be generated or NO to suppress the generation of LR statements. If the LOGREL parameter is omitted, an entry of YES is assumed. You must specify the foreign keys for a table to DB2 in order for this function of the utility to work correctly.
- **LONGNAME**
(optional) The LONGNAME parameter specifies whether the DB2 Quick Start utility should generate LX statements for the definition. An entry of YES specifies that LX statements are to be generated containing the full column name. An entry of NO specifies that LX statements should not be generated. If the LONGNAME parameter is omitted, an entry of YES is assumed.
- **DESCRIPT**
(optional) The DESCRIPT parameter specifies whether the DB2 Quick Start utility should generate D1 statements for the definition. An entry of YES specifies that D1 statements are to be generated using the Label information in the SYSCOLUMNS table. An entry of NO specifies that D1 statements should not be generated. If the DESCRIPT parameter is omitted, an entry of YES is assumed.
- **DATEFLD**
(optional) The DATEFLD parameter specifies how DB2 columns with a data type of DATE are to be handled. An entry of YES specifies that a DB2 data type of DATE should be defined as a Lilian Date (Type D) to VISION:Builder. An entry of NO specifies that a DB2 data type of DATE should be defined as a character string (Type C) with a length of 10 to VISION:Builder. If the DATEFLD parameter is omitted, an entry of NO is assumed.

SEGMENT Control Statement

Statement Syntax

```
SEGMENT  NAME= ,  
         NUMBER=,  
         LEVEL=,  
         TABLE=,  
         CREATOR=,  
         PRINT=
```

Figure 6-5 SEGMENT Control Statement Format

- **SEGMENT**
(required) **SEGMENT** is a control statement command. It identifies the control statement type. Each COMLIB segment corresponds to a specific DB2 table.
- **NAME**
(required) The **NAME** parameter assigns a name to a segment. Segment names can be from 1 to 8 characters.
- **NUMBER**
(required) The **NUMBER** parameter assigns a number that uniquely identifies the segment. Enter a number from 1 to 255. Subordinate segments must have a number larger than the parent segment and smaller than any children segments.
- **LEVEL**
(required) The **LEVEL** parameter specifies the subordination of segments. The root segment must have a level number of 1. All subordinate segments must be assigned a number from 2 to 9.
- **TABLE**
(required) The **TABLE** parameter specifies the name of the DB2 table from which the field information for this segment should be generated.
- **CREATOR**
(required) The **CREATOR** parameter qualifies the table name. Enter the authorization or creator ID for the table, or enter an asterisk (*). If an asterisk is entered, field information is generated from all tables with the specified table name, regardless of the creator ID.
- **PRINT**
(optional) The **PRINT** parameter controls the information that is printed on the summary report. To print DB2 field information, enter DB2. To print COMLIB field information, enter ANSWER. To print both DB2 and COMLIB field information, enter ALL.

NEWPAGE Control Statement

Statement Syntax

NEWPAGE

Figure 6-6 NEWPAGE Control Statement Format

- **NEWPAGE (required)** NEWPAGE is a control statement command. It causes a page eject in the summary report that is written to SYSPRINT. There are no parameters for this control statement.

Conversion Rules

Generated COMLIB File Definition

This utility generates a skeletal COMLIB file definition which must be edited and validated by the VISION:Workbench for DOS, VISION:Workbench for ISPF, or a text editor prior to its use. The following items should be considered when editing the generate relational file definition:

- **Segment key assignment** Each segment must define at least one field as the key; additional keys can be assigned if wanted.
- **Logical relationships** Logical relationships must be provided for segments defined at levels 2 to 9.
- **Segment information** Additional segment information such as segment (row) order and suppress duplication should be provided as needed.
- **Field information** Primary field name assignments can be modified if wanted (see below). Additional field information such as rounding, editing, column headings, and automatic table lookup results should be provided as needed.

COMLIB Field Name Generation

All fields within a COMLIB file definition must be assigned a unique 1- to 8-character name. This name is referred to as the primary field name.

When building a skeletal COMLIB file definition from DB2 field definitions, a unique 1 to 8-character primary field name must be assigned to each field. Since DB2 field names can be longer than 8 characters and may not be unique within the first 8 characters, DB2 quick start automatically generates a unique primary field name using the FLDPREFX parameter value on the FILEGEN statement followed by a generated field number. If the FLDPREFX parameter is omitted, the default prefix is F and the generated primary field names have the format Fnnnnnnn where nnnnnnn is a number from 0000001 to 9999999.

DB2 quick start uses the DB2 column name to generate the External Column Name specification.

Once the skeletal file definition has been created, it can be imported into the VISION:Workbench for DOS or VISION:Workbench for ISPF, and the generated primary field names can be replaced with more descriptive primary field names.

Generating COMLIB Field Information

The following table shows how the information in the DB2 SYSCOLUMNS table maps to the generated COMLIB field statements. The table on page [6-11](#) summarizes how DB2 field types and lengths are converted to COMLIB field types and lengths.

DB2 SYSCOLUMNS Field	COMLIB Field
NAME	Generates the Column Name specification.
COLTYPE	Generates the Field Type specification. See the table on page 6-11 for more information.
LENGTH	Generates the Field Length specification. See the table on page 6-11 for more information.
SCALE	Generates the Decimal Places specification. See the table on page 6-11 for more information.
KEYSEQ	Generates the Segment Key Value specification.
LABEL	Generates the Column Heading specifications.

Note: The generated COMLIB field length for a GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC string is 2*n where n is the DB2 length of the string (number of DBCS characters).

DB2		COMLIB	
Field Type	Field Length	Field Type	Field Length
INTEGER	4	Fixed (F)	4
SMALLINT	2	Fixed (F)	2
FLOAT	4	Floating Point (E)	4
FLOAT	8	Floating Point (E)	4 (Truncate)
DECIMAL	Up to 31 digits	Packed Decimal (P)	Up to 15 bytes - possible truncation
CHAR	Up to 254	Character (C)	Up to 255
VARCHAR	Up to maximum record size	Variable (V)	Up to 99H, possible truncation.
LONG VARCHAR	Up to maximum page size	Variable (V)	Up to 99H, possible truncation.
GRAPHIC	Up to 127	Character (C)	Up to 255
VARGRAPHIC	Up to maximum record size	Variable (V)	Up to 99H, possible truncation.
LONG VARGRAPHIC	Up to maximum page size	Variable (V)	Up to 99H, possible truncation.
DATE	4	Character (C)	10
TIME	3	Character (C)	8
TIMESTAMP	10	Character (C)	26

VISION:Results Quick Start

The VISION:Results™ Quick Start utility generates COMLIB file definitions from existing VISION:Results file definitions.

Input to the VISION:Results Quick Start utility is in the form of a sequential data set, PDS member, or OS/390 , CA-Panvalet or CA-Librarian copybook containing a single VISION:Results file definition. Output from the VISION:Results Quick Start utility is in the form of a single PDS member containing the converted file definition statements.

The RESULTS quick start utility can also run from VISION:Workbench for ISPF. The IMPORT Option points you to a menu for selecting the quick start utility you want to run. The subsequent panels prompt you for the information needed to run the utility. Once the information is gathered, the utility is run immediately and the output displayed for you to browse.

Flow Diagram

As shown in [Figure 7-1 on page 7-2](#), the VISION:Results Quick Start utility uses the following input data sets:

- | | |
|---------|---|
| STEPLIB | Required. STEPLIB must specify the COMLIB Release 4.5 load library. The VISION:Builder installation load library containing the parameter module M4PARAMS must also be available in the STEPLIB concatenation. |
| SYSIN | Required. SYSIN must specify an input file to be converted. It may be specified in various ways: <ul style="list-style-type: none">■ A member in a PDS (the default in the supplied JCL).■ A sequential data set.■ A VISION:Results COPY statement for data from an OS/390 copybook (SYSCOPY DD statement).■ A VISION:Results COPYP statement for data from a CA-Panvalet library (PANDD1 DD statement). |

- A VISION:Results COPYL statement for data from a CA-Librarian master file (MASTER DD statement).
- As SYSIN in-stream data.

Note: Input must be a valid VISION:Results file definition beginning with a free-form FILE statement, followed by field definition statements.

See [DD Statement Overrides on page 7-5](#) for details on SYSIN data from sources other than a PDS.

SYSCOPY	Optional. If SYSIN is to contain a COPY statement for a copybook from an OS/390 PDS member, then the SYSCOPY DD statement must be provided, pointing to the OS/390 PDS copy library.
PANDD1	Optional. If SYSIN is to contain a COPYP statement for a copybook from a CA-Panvalet library, then the PANDD1 DD statement must be provided, pointing to the CA-Panvalet library.
MASTER	Optional. If SYSIN is to contain a COPYL statement for a copybook from a CA-Librarian master file, then the MASTER DD statement must be provided, pointing to the CA-Librarian master file.

The VISION:Results Quick Start utility produces the following output data sets:

SYSPRINT	The SYSPRINT data set contains a listing of the input statements, as well as any diagnostic or informational messages.
SYS004	The SYS004 file contains the converted file definition.

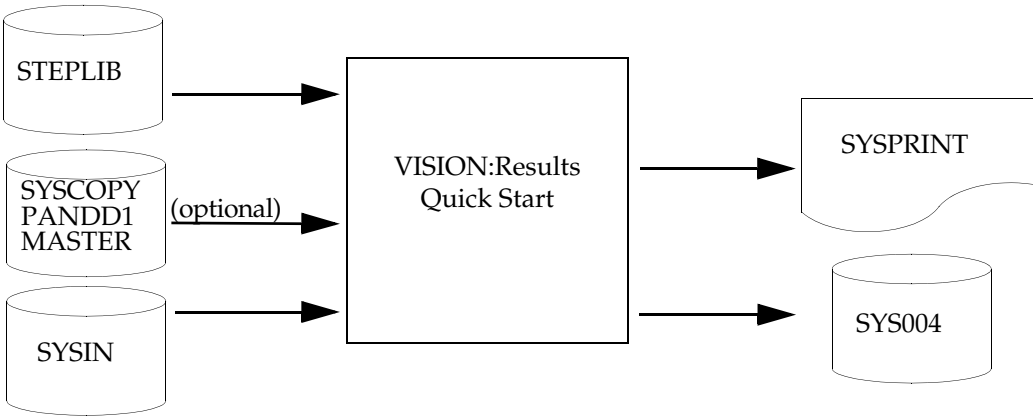


Figure 7-1 VISION:Results Quick Start Flow Diagram

Utility Execution

The VISION:Results Quick Start utility is a batch program. It can be invoked online under the Import Main Menu option of VISION:Workbench for ISPF. It can also be run in batch mode by submitting the provided JCL member, BLXRSQ#3, in the PDS data set (...PREP.JCLCNTL). This JCL contains an in-stream procedure, followed by JCL statements to execute the procedure.

[Figure 7-2 on page 7-3](#) contains a listing of the sample JCL that is provided to run the VISION:Results Quick Start utility. Review the supplied JCL carefully. At a minimum, you must make the following changes before submitting the JCL:

- Supply a JOB card.
- Supply appropriate variables for the procedure variables shown in the following table.

Variable Name	Description
RGN	Specify the region size for the utility execution. The default is 2 meg.
BLLOAD	Specify the VISION:Builder installation load library, which contains the M4PARAMS module.
DEFLIB	Specify the data set name of a PDS into which the converted file definition will be placed. The default is the Definition Processor Definition library PDS.
MEMBER	This is the output PDS member name that the converted definition will use. It is suggested that the member name match the file name of the file definition being converted.
RSLTLIB	Specify the data set name of a PDS containing the VISION:Results file definition to be converted.
RSLTDEF	Specify the PDS member name of the VISION:Results file definition to be converted.

```

/* MEMBER BLXRSQ#3                                00010000
/*****                                              00020000
/* EXECUTE THE RESULTS QUICK START UTILITY          * 00030000
/* ***** NOTE *****                          * 00040000
/* THE SYSCOPY DD STATEMENT IS USED FOR MVS COPYBOOK LIBRARIES. * 00050000
/* THE PANDD1 DD STATEMENT IS USED FOR PANVALET COPYBOOK LIBRARIES. * 00060000
/* THE MASTER DD STATEMENT IS USED FOR LIBRARIAN COPYBOOK LIBRARIES * 00070000
/*****                                              00080000
//RESLTQS PROC RGN=2M,                             00090000

```

Figure 7-2 Sample Execution JCL for the VISION:Results Quick Start Utility (Page 1

```

//          BLOAD=,                                00110000
//          DEFLIB=,                                00120000
//          MEMBER=,                                00130000
//          RSLTLIB=,                               00140000
//          RSLTDEF=,                               00150000
//CONVRT EXEC PGM=RESULTQS,REGION=&RGN              00160000
//STEPLIB DD DISP=SHR,DSN=&CLLOAD                   00170000
//          DD DISP=SHR,DSN=&BLOAD                   00180000
//SYSPRINT DD SYSOUT=*                              00190000
//*SYSCOPY DD DISP=SHR,DSN=USER.RESULTS.COPYLIB      00200000
//*PANDD1 DD DISP=SHR,DSN=USER.PANVALET.LIBRARY      00210000
//*MASTER DD DISP=SHR,DSN=USER.LIBR.MASTER          00220000
//SYS004 DD DISP=OLD,DSN=&DEFLIB(&MEMBER)           00230000
//SYSIN DD DISP=SHR,DSN=&RSLTLIB(&RSLTDEF)          00240000
//          PEND                                     00250000
//*****                                             00260000
//* FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE. BEFORE YOU * 00270000
//* RUN THIS PROCEDURE, SPECIFY:                      * 00280000
//*                                                    * 00290000
//* RGN - THE REGION SIZE. DEFAULT IS 2M.             * 00300000
//* BLOAD - THE NAME OF YOUR BUILDER LOAD LIBRARY.    * 00320000
//* DEFLIB - THE LIBRARY(PDS) TO CONTAIN THE BUILDER DEFINITIONS. * 00330000
//* MEMBER - THE PDS MEMBER NAME FOR THE CONVERTED VISION:BUILDER * 00340000
//* FILE DEFINITION IN THE DEFINITION LIBRARY.        * 00350000
//* RSLTLIB - THE PDS CONTAINING THE VISION:Results FILE * 00360000
//* DEFINITION SOURCE STATEMENTS.                     * 00370000
//* RSLTDEF - THE PDS MEMBER NAME OF THE INPUT VISION:Results * 00380000
//* FILE DEFINITION TO BE CONVERTED.                  * 00390000
//*                                                    * 00400000
//* *** N O T E ***                                   * 00410000
//*                                                    * 00420000
//* THIS PROCEDURE ASSUMES INPUT FROM A PDS MEMBER. OPTIONALLY, IT * 00430000
//* MAY ALSO COME FROM A RESULTS COPY (MVS PDS), COPYP (PANVALET), * 00440000
//* OR COPYL (LIBRARIAN) STATEMENT. IF SO, YOU MUST UN-COMMENT THE * 00450000
//* APPROPRIATE SYSCOPY (MVS PDS), PANDD1 (PANVALET), OR MASTER * 00460000
//* (LIBRARIAN) DD STATEMENT IN THE PROCEDURE, SPECIFYING THE * 00470000
//* PROPER DATA SET NAME FOR THE LIBRARY USED. PLEASE REFER TO THE * 00480000
//* MANUAL FOR DETAILS IN SETTING UP COPY SUPPORT.      * 00490000
//*****                                             00500000
//STEP01 EXEC RESLTQS,RGN=2M,                        00510000
//          BLOAD='BUILDER.BL140.LOADLIB',           00530000
//          DEFLIB='VISION.BUILDER.DEFLIB',          00540000
//          MEMBER=FILENAME,                          00550000
//          RSLTLIB='VISION.RESULTS.FILEDEFS',        00560000
//          RSLTDEF=FILENAME                          00570000

```

Figure 7-2 Sample Execution JCL for the VISION:Results Quick Start Utility (Page 2)

DD Statement Overrides

It is possible through the use of DD overrides to change the specifications for the SYSIN data set from the default of a PDS member. This may be done in one of several ways:

- To specify a sequential data set as SYSIN input, override the SYSIN after procedure invocation with the following JCL DD override:

```
//CONVRT.SYSIN DD DISP=SHR,DSN=USER.SEQ.FIELDDEF
```

- To specify the SYSIN item to be converted as in-stream data, override SYSIN after the in-stream procedure invocation as follows:

```
//CONVRT.SYSIN DD *  
(input to be converted)
```

- To specify input using OS/390 copybook support, un-comment the SYSCOPY DD statement in the procedure, and specify the data set name of a PDS containing the VISION:Results file definition to be copied. Then override the SYSIN DD statement with in-stream data as follows:

```
//CONVRT.SYSIN DD *  
COPY membername
```

- To specify input using CA-Panvalet library copybook support, un-comment the PANDD1 DD statement in the procedure, and specify the data set name of the CA-Panvalet library containing the VISION:Results file definition to be copied. Then override the SYSIN DD statement with in-stream data as follows:

```
//CONVRT.SYSIN DD *  
COPYP membername
```

- To specify input using CA-Librarian copybook support, un-comment the MASTER DD statement in the procedure, and specify the data set name of the CA-Librarian master file containing the VISION:Results file definition to be copied. Then override the SYSIN DD statement with in-stream data as follows:

```
//CONVRT.SYSIN DD *  
COPYL membername
```

Operational Characteristics

Supported Statement Types

The VISION:Results Quick Start utility processes all valid VISION:Results FILE and FIELD definition free-form statements, as well as the COPY, COPYP, and COPYL statements. Other valid VISION:Results processing and reporting statements may be included in the input, but will be ignored unless they contain errors. The VISION:Results FILE statement must be the first statement in the definition.

Converted File Definition

The VISION:Results Quick Start utility generates file definition statements of the following types:

- | | |
|----------------------------------|--|
| ■ FD (File Definition) | ■ LS (Segment Definition) |
| ■ L0 (Field Definition) | ■ Ln (Field Column Heading Definition) |
| ■ LX (Alternate Name Definition) | ■ Dn (Field Description Definition) |

Note that all text fields in the L0, Ln, LX, and Dn generated statements are scanned for the single quote ('), double quote ("), comma (,) and VISION:Builder system delimiter (from M4PARAMS) characters. If encountered, they are changed to either an underscore (_) or an "at" sign (@).

The generated FD statement is built from the VISION:Results FILE statement. The file name used is the same as the VISION:Results file name, and should be used as the "MEMBER" JCL PROC parameter of the PDS member name of the converted definition. If a key length and location are specified in the VISION:Results definition, it is saved and used to assign a key field to the converted FD.

If no key location is specified on the VISION:Results FILE statement, the first field output in the converted FD is arbitrarily designated as the key, and a diagnostic warning message is issued so that the converted FD may be inspected and changed, if necessary. Record format conversion is accomplished where possible.

If the record format from the VISION:Results definition could not be translated, a question mark (?) is inserted in the converted record format field, and a diagnostic warning message is issued so that this field may be reviewed in the converted FD.

For record and block size, the VISION:Results characters per record and characters per block fields are specified as the record size and buffer size fields, respectively, unless the record format is FIXED or ISAM, in which case the record size is set and a records per block is calculated for the FD statement.

The generated LS statement always has a name of SEGMENT1, since VISION:Results file definitions represent flat, not hierarchical, files. It is also assigned a segment number of 1 and a level number of 1.

The generated L0 statements always contain a generated field name of a 1-byte prefix (the character F), followed by a 7-digit number that is incremented by one for each field defined. The VISION:Results field name, which can be up to 50 characters in length, is used to generate an alternate name for the field (see LX statement explanation below).

If the name is greater in length than the 30 bytes allowed for alternate names, only the first 30 characters are used, and the entire 50-byte name is used as the field description (see Dn statement explanation below). When this happens, the field definition should be checked for duplicate names using the Duplicate Name Check function of the VISION:Workbench for ISPF Import option.

Field location, length, type, output edit length, and rounding information are translated directly to the L0 statement from the VISION:Results definition. Edit codes are translated as closely as possible. A field location of 4 question marks (????) will be generated if the VISION:Results definition uses 5 bytes without a leading zero to define the field location.

The generated Ln statements for column headings come from the VISION:Results field definition column heading information, if present. If no column heading is specified in the input VISION:Results definition, then the field name is used as the column heading. Since column headings are to be specified in 16-byte lines, the converted definition should be checked in case the 30-byte VISION:Results column heading has been “broken up” inappropriately during the conversion.

The generated Dn statements (field descriptions) are based upon the VISION:Results field name if it was greater than 30 bytes. If not, the column heading is used as the field description.

The generated LX statements for alternate names are based upon the first 30 characters in the VISION:Results field name. Take note that if there is more than one field that is greater than 30 bytes in the VISION:Results definition, there is the possibility of duplicate names being generated in the converted FD. The VISION:Workbench for ISPF Duplicate Name Check function should then be used upon the converted FD.

Member Naming Conventions

The member name of the converted definition is always taken from the JCL “MEMBER” PROC parameter. If this name exists in the output PDS, it will be overwritten. For ease of use with the VISION:Workbench for ISPF, the PDS name must always match the file name of the file definition being converted.

SYSPRINT Listing

The SYSPRINT listing contains the following information:

- A listing of the VISION:Results input statements.
- Error messages for any input statements that are invalid.
- Informational or warning messages about the converted FD statements.

COPYP and COPYL Support

In order to use the COPYP (CA-Panvalet copy) and COPYL (CA-Librarian copy) statements, additional setup must be done prior to execution of the VISION:Results Quick Start utility. This setup involves linking the appropriate interface routines with the VISION:Results Quick Start utility for CA-Panvalet or CA-Librarian access.

Installing the VISION:Results Quick Start Routines (Optional)

If you plan to use the VISION:Results Quick Start utility to convert definitions that are stored in a CA-Librarian or CA-Panvalet library, the appropriate interface routine must first be link edited with the VISION:Results Quick Start utility. This is accomplished by running one of the supplied JCL procedures in the WORKLIB PDS.

Link Edit CA-Librarian Support

The PDS dataset (...PREP.JCLCNTL) contains member BLXRSQ#1 to link edit support for accessing VISION:Results file definitions stored in a CA-Librarian master file with the VISION:Results Quick Start utility. [Figure 7-3](#) shows the JCL procedure. Modify the procedure variables as appropriate and run the job to activate this interface. A return code of 4, coupled with the warning message IEW0461 can be ignored.

```

/* * MEMBER BLXRSQ#1                                00010000
/*****                                              00020000
/* * LINK LIBRARIAN INTERFACE MODULES WITH RESULTS QUICK START. * 00030000
/*****                                              00040000
//LBLNK PROC LOADLIB=,                                00050000
//          LIBLOAD=                                00060000
//LINK EXEC PGM=IEWL,REGION=1M,PARM='LIST,MAP,LET,NCAL' 00070000
//SYSLIB DD DUMMY                                00080000
//SYSPRINT DD SYSOUT=*                            00090000
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))          00100000
//LIBSYS DD DISP=SHR,DSN=&LIBLOAD                  00110000
//SYSLMOD DD DISP=SHR,DSN=&LOADLIB                  00120000
//          PEND                                    00130000
/*****                                              00140000
/* * BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING * 00150000
/* * INFORMATION:                                     * 00160000
/* *                                     * 00170000
/* *   LOADLIB - NAME OF YOUR VISION:UILDER LOAD LIBRARY. * 00180000
/* *   LIBLOAD - NAME OF YOUR LIBRARIAN SYSTEM LOAD LIBRARY. * 00190000
/*****                                              00200000
//LIBLINK EXEC LBLNK,                                00210000
//          LOADLIB='BUILDER.BL140.LOADLIB',        00220000
//          LIBLOAD='LIBRARN.SYSTEM.LOADLIB'        00230000
//LINK.SYSLIN DD *                                00240000
INCLUDE LIBSYS(FAIRCLS)                            00250000
INCLUDE LIBSYS(FAIROPN)                            00260000
INCLUDE LIBSYS(FAIRREC)                            00270000
INCLUDE LIBSYS(FAIRMOD)                            00280000
INCLUDE LIBSYS(FAIRERR)                            00290000
INCLUDE LIBSYS(FAIRLOC)                            00300000
INCLUDE LIBSYS(FAIRNTE)                            00310000
INCLUDE LIBSYS(FAIRPNT)                            00320000
INCLUDE LIBSYS(FAIRSCAN)                           00330000
INCLUDE LIBSYS(FAIRSEC)                            00340000
INCLUDE SYSLMOD(DYL280LX)                          00350000
          ENTRY DYL280L                             00360000
          NAME DYL280L(R)                           00370000

```

Figure 7-3 JCL to Link CA-Librarian Support

Link Edit CA-Panvalet Support

The PDS dataset (...PREP.JCLCNTL) contains member BLXRSQ#2 to link edit support for accessing VISION:Results file definitions stored in a CA-Panvalet library with the VISION:Results Quick Start utility. [Figure 7-4](#) shows the JCL procedure. Modify the procedure variables as appropriate and run the job to activate this interface.

```

/* MEMBER BLXRSQ#2                                00010000
/******                                           00020000
/* LINK PANVALET INTERFACE MODULES WITH RESULTS QUICK START. * 00030000
/******                                           00040000
//PNLNK PROC LOADLIB=,                             00050000
//          PANLOAD=,                               00060000
//LINK EXEC PGM=IEWL,REGION=1M,PARM='LIST,MAP,LET,NCAL ' 00070000
//SYSLIB DD DUMMY                                  00080000
//SYSPRINT DD SYSOUT=*                             00090000
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))           00100000
//LIBSYS DD DISP=SHR,DSN=&PANLOAD                  00110000
//SYSLMOD DD DISP=SHR,DSN=&LOADLIB                  00120000
//          PEND                                    00130000
/******                                           00140000
/* BEFORE SUBMITTING THIS JCL, YOU MUST SPECIFY THE FOLLOWING * 00150000
/* INFORMATION:                                     * 00160000
/* *                                                * 00170000
/* * LOADLIB - NAME OF YOUR VISION:BUILDER LOAD LIBRARY. * 00180000
/* * PANLOAD - NAME OF YOUR PANVALET SYSTEM LOAD LIBRARY. * 00190000
/******                                           00200000
//PANLINK EXEC PNLNK,                              00210000
//          LOADLIB='BUILDER.BL140.LOADLIB',        00220000
//          PANLOAD='PANVALET.SYSTEM.LOADLIB'        00230000
//LINK.SYSLIN DD *                                  00240000
//          INCLUDE LIBSYS(PAM)                     00250000
//          INCLUDE SYSLMOD(DYL280PX)                00260000
//          ENTRY DYL280P                             00270000
//          NAME DYL280P(R)                           00280000

```

Figure 7-4 JCL to Link CA-Panvalet Support

Messages

Diagnostic and informational messages from the VISION:Results Quick Start utility all begin with the characters DYL-. Diagnostic messages relating to errors in the input VISION:Results file definition are documented in the VISION:Results Messages and Codes Guide. Messages relating to the converted field definition are listed below. The messages are placed in the SYSPRINT data set.

Message	Explanation
DYL-1800 - If encountered, the system delimiter of '*' will be replaced with 'x'.	Informational message. This indicates which character will replace the system delimiter character if it is encountered in a name field, field description, or column heading.

Message	Explanation
DYL1801 - Unable to convert 5-byte VISION:Results record size to VISION:Builder 4-byte format.	Warning message. This message is generated when a 5-byte VISION:Results record size is encountered. VISION:Builder allows only 4 bytes for record size, so the record size field should be checked in the converted definition. Consider leaving the record size field in FD blank, and using the buffer size field instead.
DYL1802 - Spanned record type buffer size has been estimated at 34K. Check the FD.	Warning message. Spanned record support for definitions is specified by having a buffer size of greater than 32K, whereas spanned records in VISION:Results use a field type character of S. This field should be checked in the output FD for accuracy.
DYL1803 - Unable to convert the VISION:Results record format. Check the FD.	Warning message. The record format in the VISION:Results field definition could not be translated into VISION:Builder format. The converted file definition should be checked for a valid record format specification.
DYL1804 - Unable to load module M4PARAMS. The pound sign will be assumed as the system delimiter.	Warning message. An OS/390 LOAD was issued for the M4PARAMS load module, and the module was not available in the STEPLIB concatenation. Make sure the installation load library has been made available in STEPLIB. The utility continues to run by assuming the system delimiter has not been changed from the default of #.
DYL1805 - Unable to convert 5-byte field location for field *****. Check the FD.	Warning message. VISION:Results definitions allow for 5 bytes in the field location, and VISION:Builder only allows 4 bytes. If the VISION:Results uses all 5 bytes, and the first is not a leading zero, the utility is unable to convert the location. This should be checked in the converted FD. The field name in question is inserted into the message where the asterisks appear.
DYL1806 - No key was specified: field ***** was arbitrarily designated as the key. Check the FD.	Warning message. All definitions are required to have one key designated as a key field for the file, but VISION:Results does not allow a key specification for certain file types. When this occurs, the utility arbitrarily designates the first field converted as the key field. The converted FD should be checked and updated if this designation is incorrect.

Message	Explanation
DYL1807 - The key information supplied did not match a defined field. Check the FD.	Warning message. The VISION:Results definition did have a key field length and location specified, but the utility was unable to match this with a defined field. The converted FD has not been designated with a key field. Check the converted FD and specify a key field with VISION:Workbench for ISPF.
DYL1808 - Key field ***** was designated as the key based upon the key information in the VISION:Results FD.	Informational message. The VISION:Results definition did have a key field length and location specified, and the utility was able to match this with a defined field. The key field name is replaced in the message where the asterisks occur.

Return Codes

When run in batch, the VISION:Results Quick Start utility returns to the operating system with one of the following return codes. These return codes appear as the COND CODE value in the operating system JCL log. Return codes of 0 or 4 mean that the utility continued executing until normal end of file on the SYSIN data set. A return code of 8 indicates an error from which no recovery could be made and the VISION:Results Quick Start utility terminated without creating a converted field definition.

Return Code	Explanation
0	Normal return code. No errors were encountered, and only informational messages were issued. A converted file definition in VISION:Builder format has been created.
4	Warning return code. One or more warning messages were issued during definition conversion. A converted file definition has been created, but should be closely checked for accuracy using VISION:Workbench for ISPF. The messages in the SYSPRINT listing indicate which items should be checked.
8	Error return code. Errors were detected in the syntax or content of the input VISION:Results FILE definition statements. The statements were not converted. Correct the errors on input as indicated by the messages in the SYSPRINT listing, and rerun the VISION:Results Quick Start utility.

VISION:Inquiry Quick Start

The VISION:Inquiry™ Quick Start utility generates COMLIB file definitions from existing VISION:Inquiry MAPGENs. During the utility processing, each VISION:Inquiry map that is processed becomes a separate file definition. The VISION:Inquiry map information is converted into file, segment, and field information in the file definition.

Note: If an existing member name is specified, the existing member is replaced when the new definition is saved. If a new member name is specified, then a new member is created.

The VISION:Inquiry Quick Start utility can be invoked either as an import function within the Definition Processor or as a stand-alone batch execution. The remainder of this section describes the use of VISION:Inquiry Quick Start as a batch utility.

Multiple file definitions can be generated in a single VISION:Inquiry Quick Start batch execution. Each generated file definition is stored as a separate member in the indicated source definition library. The specified file name is used as the member name when the generated definition is saved in the source definition library.

Once a file definition has been created, any additional information that is needed can be added using the VISION:Workbench for ISPF.

The INQUIRY quick start utility can also run from VISION:Workbench for ISPF. The IMPORT Option points you to a menu for selecting the quick start utility you want to run. The subsequent panels prompt you for the information needed to run the utility. Once the information is gathered, the utility is run immediately and the output displayed for you to browse.

Flow Diagram

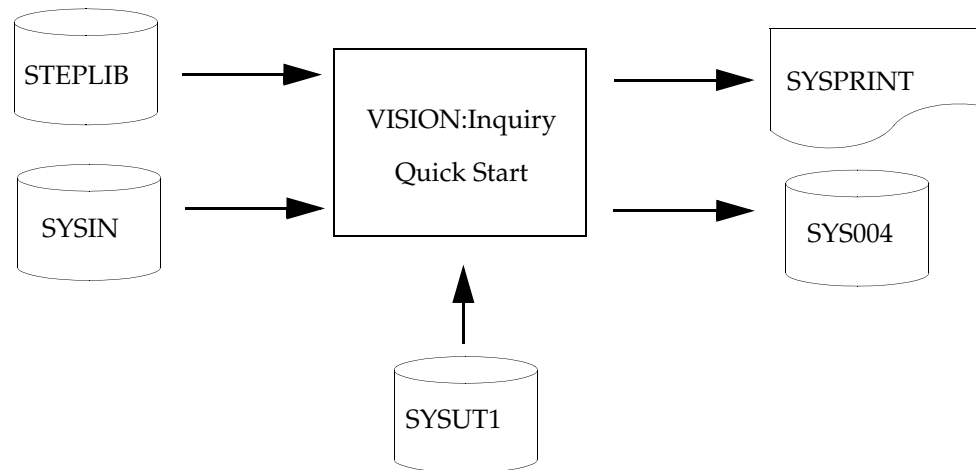


Figure 8-1 VISION:Inquiry Quick Start Flow Diagram

As shown in [Figure 8-1](#), Quick Start uses the following three input data sets:

- | | |
|---------|--|
| STEPLIB | Specifies the VISION:Builder Release 14.0 load library. |
| SYSIN | Specifies the appropriate VISION:Inquiry Quick Start control statements. |
| SYSUT1 | Contains the unloaded form of the VISION:Inquiry system database. See the VISION:Inquiry User's Guide for information on how to create the unloaded form of the system database. |

VISION:Inquiry Quick Start produces the following two output files:

- | | |
|----------|--|
| SYSPRINT | Contains a summary report on the file definition generation process. |
| SYS004 | Generated file definitions are written to the partitioned data set pointed to by the ddname SYS004. This must be a partitioned data set. |

Utility Execution

[Figure 8-2](#) contains the JCL required to execute VISION:Inquiry Quick Start. This JCL contains an in-stream procedure followed by JCL statements to execute the procedure, as well as sample control statements. The JCL may be found in the PDS data set (...PREP.JCLCNTL) member BLXINQ#1.

At a minimum, you must make the following changes before submitting this JCL:

- Supply a JOB card.
- Supply values for the procedure variables detailed in the table on page [8-3](#).
- Provide a DD statement for SYSIN. This data set contains the required VISION:Inquiry Quick Start input control statements which are used to control the generation of file definitions. See [FILEGEN Control Statement on page 8-4](#) for detailed information on these control statements.
- To conform to your shop standards, additional modifications may be necessary.

Variable Name	Description
BLLOAD	Specifies the name of the VISION:Builder Release 14.0 load library.
ULSYSDB	Specify the name of your VISION:Inquiry unloaded system database.
DEFLIB	Specify the name of your definition library.

Note: If the SYSIN data set is specified as a DUMMY or if no FILEGEN statements are specified in the input stream (empty SYSIN data set), all MAPGENs in the unloaded system database will be converted in one invocation of the VISION:Inquiry Quick Start utility.

```

/* MEMBER BLXINQ#1                                00010000
/******                                           00020000
/* UTILITY TO CONVERT VISION:Inquiry FILE DEFINITIONS INTO * 00030000
/* VISION:UILDER OR VISION:INFORM FORMAT FILE DEFINITIONS. * 00040000
/* THE VISION:Inquiry FILE DEFINITIONS MUST COME FROM AN * 00050000
/* VISION:Inquiry UNLOADED SYSTEM DATABASE FILE. SEE YOUR * 00060000
/* VISION:Inquiry TECHNICAL REFERENCE MANUAL FOR INFORMATION ON * 00070000
/* HOW TO CREATE AN UNLOADED COPY OF THE SYSTEM DATABASE. * 00080000
/* * 00090000
/* THIS UTILITY MAY ALSO BE INVOKED UNDER TSO/ISPF USING THE * 00100000
/* VISION:INFORM DEFINITION PROCESSOR IMPORT FUNCTION. * 00110000
/******                                           00120000
//INQRYQS PROC RGN=2M,                             00130000
//              BLOAD=,                             00140000
//              ULSYSDB=,                           00150000

```

Figure 8-2 Sample Execution JCL for the VISION:Inquiry Quick Start Utility (Page 1 of 2)

```
//          DEFLIB=                                00160000
//INQRYQS EXEC PGM=INQRYQS,REGION=&RGN              00170000
//STEPLIB DD DISP=SHR,DSN=&BLLOAD                   00180000
//SYSPRINT DD SYSOUT=*                             00200000
//SYSUT1 DD DISP=SHR,DSN=&ULSYSDB                   00210000
//SYS004 DD DISP=OLD,DSN=&DEFLIB                     00220000
//          PEND                                    00230000
//*****                                           00240000
/* FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE. BEFORE YOU * 00250000
/* RUN THIS PROCEDURE, SPECIFY:                      * 00260000
/*                                                    * 00270000
/* RGN - THE REGION SIZE; DEFAULT IS 2M.              * 00280000
/* BLLOAD - THE BUILDER LOAD LIBRARY.                 * 00290000
/* ULSYSDB - THE UNLOADED VISION:Inquiry SYSTEM DATABASE FILE. * 00300000
/* DEFLIB - THE VISION:INFORM DEFINITION LIBRARY.      * 00310000
//*****                                           00320000
//STEP01 EXEC INQRYQS,RGN=2M,                        00330000
//          BLLOAD='BUILDER.BL135.LOADLIB',          00340000
//          ULSYSDB='VISION.INQUIRY.UNLOADED.SYSDBASE', 00350000
//          DEFLIB='VISION.BUILDER.DEFLIB'           00360000
//SYSIN DD *                                         00370000
FILEGEN NAME=VSHPLANT,FLDPREFX=PLT                  00380000
FILEGEN NAME=SALARIES,FLDPREFX=SAL                   00390000
```

Figure 8-2 Sample Execution JCL for the VISION:Inquiry Quick Start Utility (Page 2 of 2)

FILEGEN Control Statement

The VISION:Inquiry Quick Start utility uses the following control statement as input:

FILEGEN The FILEGEN statement triggers the start of a new file definition. The NAME parameter specified on the FILEGEN statement is used as both the file name and the member name when the generated file definition is saved.

Coding Rules

When writing VISION:Inquiry Quick Start control statements, the following rules must be observed:

- Each statement must be on a single line and must begin with the FILEGEN command name.
- Each control statement may contain keyword parameters. Keyword parameters can be specified in any order and must be separated by commas. Embedded blanks are not allowed between parameters.
- The NAME keyword is required on each statement. All other keywords are optional.
- Blank lines within the SYSIN file are not accepted.
- Columns 1 through 71 of the control statement are scanned.
- Columns 72 through 80 are ignored.

Syntax

Note: If an existing member name is specified, the existing member is replaced. If a new member name is specified, a new member is created.

FILEGEN	Required. FILEGEN is a control statement command. It identifies the control statement type. The FILEGEN control statement triggers the start of a new file definition.
NAME	<p>Required. The NAME parameter specifies the name of the MAPGEN that is being converted. A map name can be 1–8 characters. If the corresponding MAPGEN is not found in the unloaded VISION:Inquiry system database, a message is printed indicating that no conversion took place.</p> <p>When the generated file definition is saved, the file name is also used as the member name unless the NEWNAME keyword is specified (see below).</p>
BUFSIZE	Optional. The BUFSIZE parameter defines the size of the buffer needed to process a logical record for IMS. Enter a number from 1 to 32760. Multiples of 1024 can be entered as nnnnK where nnnn is a number between 1 to 9999. This parameter is only used for IMS definitions and will be ignored for all other file definition types.

FLDPREFIX	<p>Optional. The FLDPREFIX parameter specifies the 1 to 3-character prefix for generating primary field names in the file definition. Primary field names are required in a file definition and must be assigned a unique 1- to 8-character name. Since VISION:Inquiry field names can be longer than 8 characters, Quick Start automatically generates a unique 8-character primary field using the FLDPREFIX value followed by a generated field number.</p> <p>If the FLDPREFIX parameter is omitted, the default prefix is F and the generated primary field names have the format Fnnnnnnnn where nnnnnnnn is a number from 00000001 to 99999999. See Field Name Generation on page 8-7 for more information.</p>
NEWNAME	<p>Optional. The NEWNAME parameter specifies a 1–8 character name which becomes the name of the converted file definition. This name is used as both the file definition name and the member name in the definition library. If this keyword is omitted, the existing name as specified using the NAME keyword is used as the file definition name and member name.</p>

Conversion Rules

Generated File Definition

This utility generates a file definition that can be edited and validated by the VISION:Workbench for ISPF.

File information	The Glossary specification may be added as appropriate and the buffer size specification may need to be adjusted for IMS files.
Segment information	Additional segment information such as segment (row) order and suppress duplication may be provided as needed. Segment key fields will have been identified during the conversion process based upon the information contained in the MAPGEN. If it is appropriate for a segment to have more than one key field specified, the key field specifications may need to be adjusted. Segment count fields will have been identified for VSAM hierarchical files as appropriate.

Field information	Primary field name assignments can be modified if wanted (see below). Additional field information such as rounding, editing, column headings, and automatic table lookup results should be provided as needed. Field description information will be filled in from the description information in the VISION:Inquiry MAPGEN. Since VISION:Inquiry MAPGENs do not contain column heading or equivalent information, no column heading information will be present in the converted file definition.
-------------------	--

Field Name Generation

All fields within a file definition must be assigned a unique 1–8 character name. This name is referred to as the primary field name.

When building a file definition from a VISION:Inquiry MAPGEN, a unique 1–8 character primary field name must be assigned to each field. Since VISION:Inquiry field names can be longer than 8 characters, VISION:Inquiry Quick Start automatically generates a unique primary field name using the FLDPREFX parameter value on the FILEGEN statement followed by a generated field number. If the FLDPREFX parameter is omitted, the default prefix is F and the generated primary field names have the format Fnnnnnnnn where nnnnnnnn is a number from 0000001 to 9999999. The original name is used as the alternate name in this case.

Once the file definition has been created, it can be refined using VISION:Workbench for ISPF. Column heading information may be added and other changes applied as appropriate.

Messages

The following table lists the VISION:Inquiry Quick Start utility messages and their explanations. The Return Code for each message resulting in a return code greater than 0 is listed with the explanation. The question mark (?) in the messages below represents a parameter which will be substituted when the message is issued.

Message	Explanation
? is an Invalid keyword.	Identifies the keyword of a FILEGEN statement that is invalid. RC=16
? Statements generated.	Information message indicating the number of statements generated while converting a file definition.
Can't open file for ?	Identifies a map for which the SYS004 file cannot be opened. RC=20
Can't open SYS004 file.	RC=20

Message	Explanation <\$paranum
Can't open SYSIN file.	RC=20
Can't open SYSUT1 file.	RC=20
Conversion aborted; Control Statement errors.	One or more errors were encountered while scanning the FILEGEN statements. RC=16
Converting File Definition ?	Information message identifying the file definition being converted.
Definition ? is for Database Type ?	The Inquiry definition is for a database type that is not supported by VISION:Builder / VISION:Inform®. RC=4
FIELD DESC. record out of sequence.	A FIELD DESCRIPTION record was encountered when not expected. RC=8
Field not found for Description ?	The related field was not found for a field description record. RC=8
FIELD record out of sequence.	A FIELD record was encountered when not expected. RC=8
File ? present but not selected.	Information message indicating that the specified definition was present on the unloaded database file, but was not selected on any FILEGEN statement.
File Definition ? not converted.	Message indicating that the specified definition was not converted, probably because the name keyword on the FILEGEN statement did not match any definition in the unloaded system database. RC=4
Insufficient memory.	There was insufficient memory available to complete the conversion of a definition. RC=12
MAP ? has too many segments.	The specified definition has more than 100 segments. RC=8
MAP ? Segment ? has too many fields.	The specified segment of the definition has more than 1000 fields. RC=8
MAPGEN Descriptor record out of sequence.	A MAPGEN Descriptor record was encountered when not expected. RC=8
Processing completed, Return code = ?	Information message indicating that the conversion program has terminated with the specified return code.
RECORD record out of sequence.	A RECORD information record was encountered when not expected. RC=8

Message	Explanation <\$paranum
Required NAME keyword/operand not found.	A FILEGEN statement was specified without the required NAME keyword or no operand was provided for the keyword. RC=16
Segment not found for field ?	The corresponding segment was not found for a field record. RC=8
SEGMENT record out of sequence.	A SEGMENT record was encountered when not expected. RC=8
Statement is not a FILEGEN statement.	A control statement was read that was not a FILEGEN statement. RC=16
Statement syntax error.	A FILEGEN statement syntax was in error. No operand following a keyword; keyword not followed by an equal sign; RC=16
This Database Type is not supported by VISION:Builder/ VISION:Inform.	Follow-on message to message "Definition ? is for Database Type ?". RC=4

Return Codes

The VISION:Inquiry Quick Start utility generates the following return codes:

Return Code	Explanation
0	Successful completion. All selected definitions were converted.
4	The definition for one or more FILEGEN statements was not converted. See the print output listing for more information about which conversions completed successfully and which definitions did not.
8	The records for a selected definition in the VISION:Inquiry unloaded system database were not in the expected sequence or an internal limit was exceeded. The conversion in progress at the time was terminated and conversion of the next definition (if any) was attempted. One or more definitions may have been converted successfully. See the print output listing for detailed information regarding the reason.
12	Insufficient memory was available to complete a conversion. One or more definitions may have been converted successfully. See the print output listing for more information.

Return Code	Explanation <\$paranum
16	A syntax error was found on one or more FILEGEN statements. Processing was terminated after all FILEGEN statements were read. No conversion was attempted.
20	The SYSIN, SYSUT1, or SYS004 file did not open successfully. Processing was terminated immediately. See the print output listing for more information.
24	The SYSPRINT file did not open successfully. Processing was terminated immediately.

ASL Examples

Code

```

; ASL Coding Example
;
CONTROL TERM, DB2 D61A INM4CALL
;
FILE REPO
FILE MASTER INPUT, NAME TEMPL, KEYS NONE
;
; END OF ASL RUN CONTROL
MAIN: PROC INFO 'Main Procedure'
;
BIRTHDAY: FIELD D 4
FEEDBACK: FIELD C 12
NAME:      FIELD V 22
;
; Combine Elements of Name into One Field
;
COMBINE LASTNAME, ' ' STORE T.NAME
COMBINE T.NAME, FIRSTNAME, MIDINIT BLANKS 1 STORE T.NAME
;
; Convert Birth Date to a Lilian Date using Picture 'YYYY-MM-DD'
;
CALL CEEDAYS USING  BRTHDATE, 'YYYY-MM-DD',
                   T.BIRTHDAY, T.FEEDBACK
;
REPORT EMPNO, T.NAME, T.BIRTHDAY
  TITLE 'Report Showing Birth Dates of all Employees'
  ORDER BY LASTNAME, FIRSTNAME, MIDINIT
  FORMAT DATEFMT DATE, HEADINGS NAME, WIDTH 80
  ITEM T.BIRTHDAY PIC P'Wwwwwwwwz, Mmm DD, YYYY'
END REPORT
;
END PROC
;
REPORT F.TODAY, F.TODAYX, F.ISDATE, F.DATE,
       F.JULIAN, F.JULANX, F.LILIAN,
       TYPE EOF
  TITLE 'Report Showing Use of All Date Flags'
  FORMAT HEADINGS NAME
END REPORT
;

```

Reports

SEP 12, 2000 Report Showing Birth Dates of all Employees PAGE 1

EMPNO	NAME	BIRTHDAY
000150	ADAMSON, BRUCE	Saturday, May 17, 1947
200340	ALONZO, ROY R	Monday, May 17, 1926
000200	BROWN, DAVID	Thursday, May 29, 1941
000050	GEYER, JOHN B	Tuesday, Sep 15, 1925
000340	GOUNOT, JASON R	Monday, May 17, 1926
000010	HAAS, CHRISTINE I	Monday, Aug 14, 1933
200010	HEMMINGER, DIAN J	Monday, Aug 14, 1933
000090	HENDERSON, EILEEN W	Thursday, May 15, 1941
000230	JEFFERSON, JAMES J	Thursday, May 30, 1935
200220	JOHN, REBA K	Friday, Mar 19, 1948
000260	JOHNSON, SYBIL V	Monday, Oct 05, 1936
000210	JONES, WILLIAM T	Monday, Feb 23, 1953
000030	KWAN, SALLY A	Sunday, May 11, 1941
000330	LEE, WING	Friday, Jul 18, 1941
000110	LUCCHESI, VINCENZO G	Tuesday, Nov 05, 1929
000220	LUTZ, JENNIFER K	Friday, Mar 19, 1948
000240	MARINO, SALVATORE M	Wednesday, Mar 31, 1954
000320	MEHTA, RAMLAL V	Thursday, Aug 11, 1932
200240	MONTEVERDE, ROBERT M	Wednesday, Mar 31, 1954
200140	NATZ, KIM N	Saturday, Jan 19, 1946
000140	NICHOLLS, HEATHER A	Saturday, Jan 19, 1946
000120	O'CONNELL, SEAN	Sunday, Oct 18, 1942
200120	ORLANDO, GREG	Sunday, Oct 18, 1942
000290	PARKER, JOHN R	Tuesday, Jul 09, 1946
000270	PEREZ, MARIA L	Tuesday, May 26, 1953
000160	PIANKA, ELIZABETH R	Tuesday, Apr 12, 1955
000070	PULASKI, EVA D	Tuesday, May 26, 1953
000130	QUINTANA, DOLORES M	Tuesday, Sep 15, 1925
000280	SCHNEIDER, ETHEL R	Saturday, Mar 28, 1936
200280	SCHWARTZ, EILEEN R	Saturday, Mar 28, 1936
000180	SCOUTTEN, MARILYN S	Monday, Feb 21, 1949
000310	SETRIGHT, MAUDE F	Tuesday, Apr 21, 1931
000250	SMITH, DANIEL S	Sunday, Nov 12, 1939
000300	SMITH, PHILIP X	Tuesday, Oct 27, 1936
000100	SPENSER, THEODORE Q	Tuesday, Dec 18, 1956
200310	SPRINGER, MICHELLE F	Tuesday, Apr 21, 1931
000060	STERN, IRVING F	Saturday, Jul 07, 1945
000020	THOMPSON, MICHAEL L	Monday, Feb 02, 1948
000190	WALKER, JAMES H	Wednesday, Jun 25, 1952
200330	WONG, HELENA	Friday, Jul 18, 1941
200170	YAMAMOTO, KIYOSHI	Friday, Jan 05, 1951
000170	YOSHIMURA, MASATOSHI J	Friday, Jan 05, 1951

09/12/00 Report Showing Use of All Date Flags PAGE 1

TODAY	TODAYX	ISDATE	DATE	JULIAN	JULANX	LILIAN
091200	09122000	20000912	SEP 12, 2000	00256	2000256	09/12/2000

Listing

```
SEP 15, 2000 17.18.13                                     PAGE 1
*****
* VISION:Builder 4400 (OS/390 - 14.0) *
*      COPYRIGHT 2001      *
* COMPUTER ASSOCIATES INTERNATIONAL, INC. *
*****

VISION:Builder 14.0 Dev.          ENABLED FOR IBM LANGUAGE ENVIRONMENT          BUILD STAMP = 100249,20:27:02.

=====
=
=          INSTALLATION  PARAMETERS  (M4PARAMS, MARKLIBP)
=
= SYSTEM DELIMITER: #          PAGE HEIGHT: 66          MALIST WIDTH: 132          DEF WIDTH OF PAGE: 0
=
= AUTO GRAND: N          HEADING CHAR: -          SUBTITLE REPEAT: N          INVALID FIELD: *
=
= MISSING FIELD: -          NON-EDIT FIELD: +          PERCENT CHAR: %          LEFT SEPARATOR: (
=
= RIGHT SEPARATOR: )          SINGLE SEPARATOR: ,          SOURCE SPACING: 1          PRINT MESSAGES: Y
=
= CONSOLE MESSAGES: N          MAREPO BLOCKSIZE: 4,096          INPUT I/O BUFFERS: 2          OUTPUT I/O BUFFERS: 1
=
= SNGL-STEP STORAGE: 8,192          SNGL-STEP SORTSIZE: 524,288          DIGIT SELECT CHAR: 9          ZERO SUPPRESS CHAR: Z
=
= CURRENCY CHAR: $          PLUS CHAR: +          MINUS CHAR: -          CHECK PROTECT CHAR: *
=
= DECIMAL CHAR: .          GROUPING CHAR: ,          PRIMARY PLOT CHAR: X          SECONDARY PLOT CHAR:*
=
= FIT PLOT CHAR: .          HORIZONTAL AXIS: _          HORIZONTAL HASH: |          VERTICAL AXIS: |
=
= VERTICAL HASH: -          MINUTES/HOUR: 60          SECONDS/MINUTE: 60          TIME DELIMITER: HH:MM:SS
=
= DATE FORMAT:   MMM DD, YYYY          TODAY FORMAT+DELIM: MM/DD/YY          ISDATE DELIMITER: YYYY-MM-DD          JULIAN DELIMITER: YY.DDD
=
= SORT PROGRAM CODE: 2          MINCORE VALUE: 12 K          ALT MALIST WIDTH: 132          ALT DEF W/OF PAGE: 0
=
= MAX LINES OF TRACE: 1,024          ITEM TRACKING: 0          SUPPRESS NDS REPT?: N          DEFAULT MAXGETMN: 1,024 K
=
= CONDITION CODE 1: 0          CONDITION CODE 2: 4          CONDITION CODE 3: 0          CONDITION CODE 4: 16
=
= ISAM INDEX INCORE: N          MALIB BLKG FACTOR: 0          MALIB RESERVE: 0          MALIB COMPONENT: C4.5 B/V
=
= DEFAULT F/P AMODE: 31          MALIB COMPRESSION?: Y          MALIB COMPRESS MIN: 507
=
=====

SEP 15, 2000 17.18.13                                     PAGE 2
CONTROL TERM
FILE MASTER INPUT, NAME ALL80, KEYS NONE, IO_PLUGIN M4PDSIN
FILE SUBF1 NAME SUBOUT1

=====
=
=          FILE SUMMARY
=
= JOBNAME: REDJ004P          STEPNAME: M4RUN
=
= FILE NAME: M4OLD          DSNAME: REDJ004.CLIST
= FILE NAME: M4SUBF1          DSNAME: REDJ004.REDJ004P.JOB03283.D0000104.?
= FILE NAME: M4LIB          DSNAME: REDJ004.DEVEL2.M4LIB
=
=====

SEP 15, 2000 17.18.13                                     PAGE 3
*****
* PROC NAME - MAIN *
* INPUT STREAM PROCEDURE *
*****

MAIN: PROC
EXTRACT FILE SUBOUT1, ENTIRE 0
END PROC

SEP 15, 2000 17.18.13                                     PAGE 4
** MK4UI06 TYPE 0 USER I/O MODULE M4PDSIN SUCCESSFULLY LOADED VIA LOAD (SVC 8). *****
```

```
SEP 15, 2000 17.18.13
** MK4W204 TYPE 0 NUMBER OF MESSAGES PRINTED IS 1.
M4OLD - 7991 RECORDS INPUT
M4SUBF1 - 7991 RECORDS OUTPUT
M4INPUT - 6 RECORDS INPUT
M4LIST - 78 RECORDS OUTPUT
15 TRACKS ASSIGNED TO M4LIB -- 6 TRACKS NOT FULL. LIBRARY DIRECTORY BLOCKING FACTOR IS 694.
** MK4W209 TYPE 0 1019144 BYTES OF MAIN STORAGE UNUSED DURING DECODING PHASE
1019448 BYTES OF MAIN STORAGE UNUSED DURING COMPILATION PHASE
987136 BYTES OF MAIN STORAGE UNUSED DURING PROCESSING PHASE
```

PAGE 5

Index

Symbols

\$COBOL, 5-7
\$ECOBOL, 5-7

A

ASL (Advanced Syntax Language) feature, 2-1
ASL EXTRACT statement
 four variations of, 2-11
ASL report statements, 2-8
ASL run control statements, 2-3
ASL statements, 2-8

B

benefits, 1-3
books, 1-6
 discontinued books, 1-9
BUFSIZE, 5-9, 6-6

C

CA-Librarian support
 VISION:Results, 7-8
CA-Panvalet support
 VISION:Results, 7-9
COBOL and VISION:Builder, 4-7
COBOL Quick Start, 5-1
 coding rules, 5-7

 conversion rules, 5-11
 execute, 5-3
 flow diagram, 5-2
 generated COMLIB file def, 5-11
 JCL, 5-3

column heading specifications, 5-12

COLUMN name, 2-10

COMLIB

 Generating field info, 6-10

COMLIB Library Utilities, 1-1

contacting Computer Associates, web page, 1-13

COPY statement, 2-6

COPYCOBOL, 5-10

COPYLCOBOL, 5-10

COPYPCOBOL, 5-10

courses for VISION:Builder, 1-11

CREATOR, 6-8

customer-requested enhancements, 2-15

D

data field and record processing, 4-3

DATEFLD, 6-7

DB2 Quick Start, 6-1

 coding rules, 6-5

 conversion rules, 6-9

 convert fields to COMLIB, 6-10

 execute, 6-2

 flow diagram, 6-2

- generated COMLIB file def, 6-9
- JCL, 6-2
- DB2CNTL, 6-4, 6-5
- DB2PLAN, 6-5
- DB2SYS, 6-5
- DBRM, 6-2
- definition processing, 4-6
- delimited data output enhancements, 2-16
- DESCRIPT, 6-7
- discontinued books, 1-9
- documentation, 1-6
 - discontinued books, 1-9

E

- educational and professional services, 1-11
- END statement, 2-10
- enhancements
 - HTML primary document name change, 2-18
- enhancements, 2-1
 - ASL report statements, 2-8
 - ASL run control statements, 2-3
 - customer-requested, 2-15
 - delimited data output, 2-16
 - extracted data files following the sort, 2-14
 - HFS output for HTML report, 2-18
 - long field names, 2-13
 - no limit on extracted data files, 2-13
 - PL/I-like varchar output, 2-15
- environment, 1-4
- extended subfiles, 2-14
- external column name, 6-10
- extracted data files
 - following the sort, 2-14
 - no limit, 2-13

F

- features, 1-2

- file manipulation, 4-4
- FILE MASTER statements, 2-3
- FILE statements
 - additional keywords, 2-4
- FILEGEN, 5-7, 5-8, 6-4, 6-6
- FIXED keyword, 2-12
- fixed-form-syntax statements, 2-1
- Flat file, 1-1
- FLDNAME, 6-7
- FLDPREFIX, 5-9, 6-6
- functional specifications, 2-2

G

- general report semantics, 2-10

H

- HEADING, 6-7
- HFS output for HTML report, 2-18
- HTML primary document name change, 2-18

I

- INCLUDE statement, 2-12
- installation instructions
 - VISION:Results Quick Start, 7-8
- ITEM statement, 2-10

J

- JCL
 - COBOL Quick Start, 5-3
 - DB2 Quick Start, 6-2
 - VISION:Inquiry Quick Start, 8-3
 - VISION:Results Quick Start, 7-3

K

- KEYNAME parameter, 2-7

L

LEVEL, 5-10, 6-8
LOGREL, 6-7
long field names, 2-1, 2-13
LONGNAME, 6-7

N

NAME, 5-8, 5-9, 6-6, 6-8
NEWPAGE, 6-4, 6-9
NUMBER, 5-10, 6-8

P

performance tuning, 4-7
PL/I-like varchar output, 2-15
PRINT, 6-8
PROC statement, 2-12

R

RECBK, 5-8
RECSIZE, 5-8
Report Blocks, 2-8
REPORT statement
 built-in functions, 2-9
reporting from special data files, 4-2
requirements, 1-4

S

SECTION statement, 2-10
SEGMENT, 5-7, 5-9, 6-4, 6-8
SEGMENT and SSA clause, 2-7
SEGMENT and WHERE clause, 2-7
specialized report formats, 4-1
SQL clause, 2-6
STEPLIB, 5-2, 6-2

SYS004, 5-2, 6-2
SYSCOPY, 5-2
SYSIN, 5-2, 6-2
SYSPRINT, 5-2, 6-2

T

TABLE, 6-8
table processing, 4-7
technical support, contacting Computer Associates, 1-13
TYPE, 5-8

V

VISION:Builder
 ASL Run Control statements, 2-3
 conventions for specifying syntax of commands and functions, 2-2
 enhancements, 2-1
VISION:Builder Engine, 1-1
VISION:Builder system, 1-1
VISION:Inquiry Quick Start, 8-1
 coding rules, 8-5
 control statement, 8-4
 execute, 8-3
 JCL, 8-3
 VISION:Inform field name generation, 8-7
VISION:Results Quick Start, 7-1
 COPYP and COPYL support, 7-7
 DD statement overrides, 7-5
 flow diagram, 7-1
 installation, 7-8
 JCL, 7-3
 member naming conventions, 7-7
 messages, 7-9
 return codes, 7-11
 support statement types, 7-5
 SYSPRINT listing, 7-7
VISION:Workbench for DOS, 1-1
VISION:Workbench for ISPF, 1-1

W

web page

Computer Associates, 1-13